Deep and greedy kernel methods: Algorithms, analysis and applications

Von der Fakultät Mathematik und Physik sowie dem Stuttgarter Zentrum für Simulationswissenschaften der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

vorgelegt von

Tizian Wenzel

aus Nagold

Hauptberichter:	Prof. Dr. Bernard Haasdonk
Mitberichter:	Prof. Ph.D. Elisabeth Larsson
	Prof. Dr. Armin Iske
Tag der mündlichen Prüfung:	29. September 2023

Institut für Angewandte Analysis und Numerische Simulation

Universität Stuttgart

2023

In memory of my father

Acknowledgements

While the scientific work was done by me, this was enabled also by other people: I would like to thank my supervisor Prof. Dr. Bernard Haasdonk for his guidance into the scientific world, stimulating questions, motivating words and feedback all the way. Special thanks in particular to the scientific freedom he gave me on this journey, which allowed me to also delve into and deliver in related topics and learn about other topics. Furthermore, I would like to thank Dr. Gabriele Santin for plenty in depth discussions, ideas and useful references. The discussions were always fun to me and they served as a catalyst to come up with novel ideas and proof strategies. I also want to thank Prof. Dr. Robert Schaback for infrequent but motivating conversation and intriguing open research questions, which challenged me to solve them. Moreover, I want to thank Prof. Dr. Jens Wirth for listening to my analysis related thoughts, discussing them and pointing me to relevant work in the area of analysis.

My everyday work was also supported by my colleagues: First I would like to thank the team of the Institute of Applied Analysis and Numerical Simulation (IANS). In particular, I want to thank Patrick Buchfink for helping me with my questions of any kind – even if he did not have an answer himself, he always had a good suggestion whom to talk to. Furthermore I thank Daniel Winkle for sharing the same passion for kernel methods, and listening and discussing intriguing questions revolving around kernel approximation at any daytime. Great administrative support was provided by Brit Steiner and the administrators Jörg Hörner and Claus-Justus Heine, which always had time to answer my organizational and computer related questions and helped me to find a solution. Furthermore, I also like to mention Prof. Dr. Steinwart and the team from the Institute for Stochastics and Applications (ISA), in particular David Holzmüller, Simon Fischer and Thomas Hamm for many interesting discussions, works and pointers to relevant literature. Special thanks goes to Martin Tschöpe and Juliane Heitkämper as well as Manuel Bihler for sharing also non-scientific advices along the way.

My PhD research time was also that fruitful and exciting due to two stimulating research stays abroad: First, I thank Prof. Dr. Johan Suykens and the whole team of ESAT, and in particular Arun Pandey for discussing and learning a lot about deep kernel machines and the broader use of kernels. Second, I thank Prof. Dr. Lorenzo Rosasco and all the colleagues in MalGa for great discussions and joint work on the scaling of kernel methods and my improved understanding of them for machine learning. I happily remember the great conversations, stimulating ideas and challenging questions and how they helped me to extend my scientific point of view. Both of these research stays had been a great time, where I did not only learn a lot scientifically, but also grew considerably personally – I'm very grateful for these experiences and to the people I met and the friends

I made.

For ideological and financial support, I acknowledge the support of the German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes): Due to it, I had the possibility to meet plenty of fascinating people and have intriguing conversations and experiences, for which I am very grateful. For this I especially want to thank my friend Eduard Neu for motivating me to apply for this award and support, and providing helpful advice while doing so.

Finally, I want to thank my family and friends for their support and for the great time I had during this period.

Abstract

Machine learning and in particular deep learning techniques are nowadays state of the art and used in everyday life. Beyond, so-called kernel methods are another class of machine learning methods, which however can be seen as *shallow methods*. Nevertheless they are still used for approximation theory, machine learning and surrogate modeling among others, especially due to their appealing mathematical framework. This thesis deals with these kernel methods, in particular *greedy* kernel methods and advances in terms of algorithms, analysis and possible applications – in particular several ways of introducing depth into kernel methods are presented in order to benefit from the advantages of deep models.

The first main part of the thesis, Chapter 3, focuses on algorithms and analysis from the theoretical point of view, extending the theory and analysis for *shallow* kernel methods. The main vehicles for these improvements were provided by abstract tools from approximation theory, which were applied to kernel approximation by leveraging and extending a convenient connection, which was recently established in the literature. Notably it was possible to unify greedy kernel algorithms in a joint framework and prove strong target data-dependent convergence rates for greedy kernel approximation for the first time.

The second main part of the thesis, Chapter 4, focuses on algorithms and analysis as well, however from the practical point of view by introducing two novel *deep* kernel approaches. Firstly, structured deep kernel networks (SDKNs) are presented, which are constructed by using simple classes of kernels in a multilayer network setup, akin to neural networks. Theoretical analyis explores their approximation properties in different asymptotic regimes, where efficient constructions are feasible by leveraging "flat limit" properties of kernels. Secondly, two-layered kernels (2L) are presented, which can be seen as hyperparameter optimized kernels. An efficient optimization strategy is given, which allows to use these two-layered kernels for subsequent kernel approximation tasks, thus introducing the 2L-VKOGA algorithm.

The third main part of the thesis, Chapter 5, deals with the application part of this thesis and shows the practical use and benefits of the newly introduced methods. A first application addresses the challenging task of predicting closure terms for the simulation of turbulent flows, for which SDKNs were employed. The second application relates to modeling the human spine, where greedy kernel models serve as surrogate models for intervertebral discs within a larger model.

Kurzfassung

Maschinelles Lernen und insbesondere Methoden des "Deep Learning" (tiefes Lernen) sind heutzutage "state of the art" und werden im Alltag genutzt. Andererseits gibt es sogenannte Kern-Methoden, eine weitere Klasse von maschinellen Lernverfahren, die jedoch als *flache Methoden* angesehen werden können. Dennoch werden sie nach wie vor in der Approximationstheorie, für maschinelles Lernen und Surrogatmodellierung eingesetzt, insbesondere aufgrund ihrer ansprechenden mathematischen Theorie. Diese Arbeit beschäftigt sich mit diesen Kern-Methoden, vor allem mit *greedy* Kern-Methoden und präsentiert Fortschritte im Bezug auf Algorithmen, Analysis und mögliche Anwendungen – insbesondere werden verschiedene Möglichkeiten zur Einführung von "depth" (Tiefe) in Kern-Methoden vorgestellt, um von den Vorteilen tiefer Modelle zu profitieren.

Der erste Hauptteil der Arbeit, Kapitel 3, konzentriert sich auf Algorithmen und Analysis aus theoretischer Sicht und erweitert die Theorie und Analyse flacher Kern-Methoden. Das Hauptwerkzeug für diese Verbesserungen wurde durch abstrakte Werkzeuge aus der Approximationstheorie bereitgestellt, welche auf die Kern-Approximation angewendet werden konnten, indem eine kürzlich in der Literatur etablierte Verbindung genutzt und erweitert wurde. Insbesondere war es möglich, greedy Kern-Algorithmen in einem gemeinsamen Rahmen zu vereinheitlichen und erstmals starke Konvergenzraten für von den Zieldaten abhängige greedy Kern-Approximation zu beweisen.

Der zweite Hauptteil der Arbeit, Kapitel 4, konzentriert sich erneut auf Algorithmen und Analysis, jedoch aus praktischer Sicht, indem zwei neue Ansätze für tiefe Kern-Methoden vorgestellt werden. Erstens werden strukturierte tiefe Kern-Netzwerke (SDKNs) vorgestellt, die durch die Verwendung einfacher Klassen von Kernen in einem mehrschichtigen Netzwerk aufgebaut werden, ähnlich wie neuronale Netzwerke. Die theoretische Analyse erforscht ihre Approximationseigenschaften in verschiedenen asymptotischen Regimen, bei denen effiziente Konstruktionen durch Ausnutzung des "flat limits" (flacher Grenzfall) der Kerne möglich sind. Zweitens werden zweischichtige Kerne (2L) vorgestellt, die als hyperparameteroptimierte Kerne betrachtet werden können. Es wird eine effiziente Optimierungsstrategie präsentiert, die es ermöglicht, diese zweischichtigen Kerne für nachfolgende Kern-Approximationsaufgaben zu verwenden und somit den 2L-VKOGA Algorithmus einführt.

Der dritte Hauptteil der Arbeit, Kapitel 5, behandelt den Anwendungsteil dieser Arbeit und zeigt den praktischen Nutzen und die Vorteile der neu eingeführten Methoden auf. Eine erste Anwendung betrifft die anspruchsvolle Aufgabe der Vorhersage von Schließungstermen für die Simulation von turbulenten Strömungen, für die SDKNs eingesetzt wurden. Die zweite Anwendung bezieht sich auf die Modellierung der menschlichen Wirbelsäule, bei der greedy Kern-Modelle als Surrogatmodelle für Bandscheiben innerhalb

eines größeren Modells genutzt werden.

Contents

Chapter 1

Introduction

1.1 Motivation

Some of the most prominent advances in science of the last decades were achieved in machine learning and deep learning, which is already ubiquitous in everyday life. Important examples have been achieved in the field of computer vision and natural language processing, e.g. self-driving cars or chatbots, especially recently ChatGPT[1]. This surge of deep learning methods was enabled by an unprecedented availability of data and computational power in recent years. Those methods frequently rely on artificial neural networks, which are the state-of-the-art machine learning tools in several applications.

Besides neural networks, another direction within machine learning can be summarized as kernel methods, which was one of the main machine learning directions before. Kernel methods are usually non-parametric models revolving around the use of a kernel, which is a symmetric function that allows, in simple terms, to implicitly turn linear algorithms into nonlinear ones. This usually gives rise to a convex optimization problem, which can be solved efficiently – however the method is rather applicable to comparably small training data sets due to an at least challenging $\mathcal{O}(n^2)$ scaling in the dataset set size n. In fact, there are also several intriguing connections between neural networks and kernel methods, e.g. in different scaling limits of the neural network: On the one hand, the so-called NNGP (neural network Gaussian process) describes the initialization of neural networks in the infinite width limit [62, 124, 131]. On the other hand, the NTK (neural tangent kernel) can be used to describe the training behavior of neural networks under gradient descent in the infinite width limit [53], which is therefore also called "kernel regime". In general, kernel methods are still state of the art in several use cases, e.g. on small data set tasks, they were shown to outperform neural networks [6]. Furthermore, kernel methods are also used in various other fields of mathematics such as approximation theory or numerical analysis, which is due to their rich mathematical background.

This motivates the further research in and development of kernel methods. A very classical result from the theory of kernels is the kernel representer theorem [56, 101, 110],

[1] `https://chat.openai.com/`

that states that a scalar-valued kernel model looks like

$$s_n(\cdot) = \sum_{j=1}^{n} \alpha_j^{(n)} k(\cdot, x_j), \quad \alpha_j^{(n)} \in \mathbb{R},$$

where k is the kernel, x_j are given input data and $\alpha_j^{(n)}$ are coefficients that can be frequently calculated explicitly based on given data, see also Theorem 6. A classical task when striving for an efficient kernel model is to balance between accuracy (e.g. a classification rate or an L^∞ error) of the kernel model and its computational cost (e.g. training and prediction times). As the coefficients can be computed, the only choices to be made is about the kernel k and the input data $\{x_j\}_{j=1}^n$. For the choice of the used input data, we investigate greedy methods that subselect a suitable amount of *centers* from the given input data, as elaborated in Chapter 3. This constitutes the more theoretical, mathematical analysis part of this thesis. For the choice of a suitable kernel, we focus on data adapted and deep kernel models and introduce two approaches, namely *Structured Deep Kernel Networks* (SDKNs) as well as *two-layered kernels* (2L) in conjunction with greedy algorithms (2L-VKOGA), see Chapter 4. These chapters constitute the more algorithmic part of the thesis.

Advancements in theoretical results and algorithmic aspects are of particular interest if they also allow for practical improvements: Therefore, these novel insights are used for surrogate modeling purposes, where kernel model are used as sparse and accurate models. This is presented in Chapter 5.

1.2 Structure of this thesis

This thesis is structured as follows:

Chapter 2 provides the background information required for the subsequent chapters of this thesis including basic knowledge about kernels, kernel approximation, greedy algorithms and deep models. Like this also the common notation for the subsequent sections is set.

Chapter 3 starts with an overview of the state of the art and presents some necessary in detail background information. Subsequently, the novel scientific contributions are presented: First, a utility link between abstract analysis and kernel analysis is extended. This strong link is then used in two directions: First, it is leveraged to prove a kind of stability result, namely how $L^\infty(\Omega)$ convergence rates can be transferred to subsets $\tilde{\Omega} \subset \Omega$. Secondly, it is leveraged to provide a full analysis for target data-dependent greedy kernel algorithms. Third, an overview of the same analysis for the solution of PDEs is presented. Chapter 4 starts by discussing some state-of-the-art methods for deep kernel learning. Then, two newly developed approaches are presented: First, structured deep kernel networks (SDKNs) are presented in Section 4.2, which combine classes of simple kernels with a network structure. Secondly, two-layered kernels (2L) are introduced in Section 4.3, which can be seen as a generalization of hyperparameter tuned kernels, where the tuning is done by a machine-learning optimization. Chapter 5 presents and discusses applications of the previously shown analysis and the introduced algorithms for surrogate modeling. For this, challenges from turbulence and human spine modeling were chosen. Chapter 6 provides a concise overview on the software developed during the course of this

studies, while Chapter 7 summarizes the work with a conclusion and the appendices A and B collect several deferred proofs.

1.3 Publications

During the time of this PhD studies, I was working on several publications:

- Published:

  1. T. Wenzel, G. Santin, and B. Haasdonk. A novel class of stabilized greedy kernel approximation algorithms: Convergence, stability and uniform point distribution. *Journal of Approximation Theory*, 262:105508, 2021
  2. P. Gavrilenko, B. Haasdonk, O. Iliev, M. Ohlberger, F. Schindler, P. Toktaliev, T. Wenzel, and M. Youssef. A full order, reduced order and machine learning model pipeline for efficient prediction of reactive flows. In *Large-Scale Scientific Computing*, pages 378–386. Springer International Publishing, 2022
  3. B. Haasdonk, T. Wenzel, G. Santin, and S. Schmitt. Biomechanical Surrogate Modelling Using Stabilized Vectorial Greedy Kernel Methods. In F. J. Vermolen and C. Vuik, editors, *Numerical Mathematics and Advanced Applications ENUMATH 2019*, pages 499–508, Cham, 2021. Springer International Publishing
  4. T. Wenzel, M. Kurz, A. Beck, G. Santin, and B. Haasdonk. Structured Deep Kernel Networks for Data-Driven Closure Terms of Turbulent Flows. In *Large-Scale Scientific Computing*, pages 410–418, Cham, 2022. Springer International Publishing
  5. T. Wenzel, G. Santin, and B. Haasdonk. Analysis of target data-dependent greedy kernel algorithms: Convergence rates for f-, f· P-and f/P-greedy. *Constructive Approximation*, 57(1):45–74, 2023
  6. B. Haasdonk, H. Kleikamp, M. Ohlberger, F. Schindler, and T. Wenzel. A New Certified Hierarchical and Adaptive RB-ML-ROM Surrogate Model for Parametrized PDEs. *SIAM Journal on Scientific Computing*, 45(3):A1039–A1065, 2023
  7. T. Wenzel, F. Marchetti, and E. Perracchione. Data-driven kernel designs for optimized greedy schemes: A machine learning perspective. *SIAM Journal on Scientific Computing*, 46(1):C101–C126, 2024
  8. M. Hammer, T. Wenzel, G. Santin, L. Meszaros-Beller, J. P. Little, B. Haasdonk, and S. Schmitt. A new method to design energy-conserving surrogate models for the coupled, nonlinear responses of intervertebral discs. *Biomechanics and Modeling in Mechanobiology*, pages 1–24, 2024
  9. F. Döppel, T. Wenzel, R. Herkert, B. Haasdonk, and M. Votsmeier. Goal-Oriented Two-Layered Kernel Models as Automated Surrogates for Surface Kinetics in Reactor Simulations. *Chemie Ingenieur Technik*, 2024

- Accepted:

  10. T. Wenzel, B. Haasdonk, H. Kleikamp, M. Ohlberger, and F. Schindler. Application of Deep Kernel Models for Certified and Adaptive RB-ML-ROM Surrogate Modeling. *arXiv preprint arXiv:2302.14526*, 2023. Accepted for LSSC 2023 proceedings
  11. T. Wenzel, G. Santin, and B. Haasdonk. Stability of convergence rates: Kernel interpolation on non-Lipschitz domains. *arXiv preprint arXiv:2203.12532*, 2022. Accepted for publication in IMA Journal of Numerical Analysis

- Preprints:

  12. T. Wenzel, D. Winkle, G. Santin, and B. Haasdonk. Adaptive meshfree solution of linear partial differential equations with PDE-greedy kernel methods. *arXiv preprint arXiv:2207.13971*, 2022
  13. T. Wenzel, G. Santin, and B. Haasdonk. Universality and Optimality of Structured Deep Kernel Networks. *arXiv preprint arXiv:2105.07228*, 2021
  14. G. Santin, T. Wenzel, and B. Haasdonk. On the optimality of target-data-dependent kernel greedy interpolation in Sobolev Reproducing Kernel Hilbert Spaces. *arXiv preprint arXiv:2307.09811*, 2023
  15. T. Wenzel. Sharp inverse estimates for radial basis function interpolation: One-to-one correspondence between smoothness and approximation rates. *arXiv preprint arXiv:2306.14618*, 2023

The articles 5, 1, 11 and 12 constitute the mostly theoretical analysis on kernel approximation. The work of the articles 5 and 11 is presented in Section 3.5 respective Section 3.4, while an overview of article 12 is briefly given in Section 3.6. Article 1 is not presented in this thesis, as it is mostly work resulting from the author's master thesis [114]. The articles 13 and 7 introduce two deep kernel algorithms and are presented in Section 4.2 and Section 4.3. Applications of these works are shown in the articles 4, 2, 3 and 6. Some details of the articles 4 and 3 are elaborated in Section 5.1 and Section 5.2, while article 2, 6, 15, 14, 9 are not covered in this thesis.

Chapter 2

Background

The following sections review the required background for the topics researched in this thesis: Kernels, greedy kernel algorithms and deep models. It is not meant to cover all the available details, but pointers to the literature with further background information are provided. Most of this content is covered in [33, 34, 105, 112], while for Section 2.6 we additionally point to [42].

2.1 Kernels

Given a nonempty set Ω, a real-valued *kernel* is a symmetric function $k : \Omega \times \Omega \to \mathbb{R}$. In general the set Ω can be quite arbitrary, but this thesis revolves around kernels defined on subsets of the Euclidean space, i.e. $\Omega \subseteq \mathbb{R}^d, d \in \mathbb{N}$. Furthermore, it is also possible to consider complex-valued kernels or matrix-valued kernels, but this is neither the focus. We remark that parts of the results presented in the following chapters can also be easily generalized (under some mild assumptions) to complex-valued or matrix-valued kernels.

A key quantity in the following is the *kernel matrix* A_{X_n}, which is defined for any $X_n \subset \Omega$ as $(A_{X_n})_{ij} := k(x_i, x_j)$, $1 \leq i, j \leq n$. An important characterization of kernels is given by its positive definiteness, and we focus on *(strictly) positive definite ((s)pd) kernels*: A kernel is positive definite, iff its kernel matrix is positive semi-definite for any set $X_n, n \in \mathbb{N}$ of pairwise distinct points. A kernel is strictly positive definite, iff its kernel matrix is positive definite for any set $X_n, n \in \mathbb{N}$ of pairwise distinct points. For the following we assume $\Omega \subset \mathbb{R}^d$ to be bounded and focus on strictly positive definite real valued kernels. This neglects conditional positive definite (cpd) kernels, however much of the theory can also be extended, with some care, to those kernels. Indeed, every cpd kernel has an associated (s)pd kernel [98].

From the application point of view, translational invariant kernels are of interest because they can be implemented and evaluated easily. They are defined on $\mathbb{R}^d \times \mathbb{R}^d$: A kernel is called translational invariant iff

$$k(x, y) = k(x + z, y + z) \quad \forall x, y, z \in \mathbb{R}^d,$$

and is called radial iff

$$k(x, y) = k(z, w) \quad \forall x, y, z, w \in \mathbb{R}^d \text{ such that } \|x - y\|_2 = \|z - w\|_2.$$

The class of translational invariant and the class of radial kernels can be also characterized with the following proposition:

Proposition 1. *A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is translational invariant, iff there exists a $\Phi : \mathbb{R}^d \to \mathbb{R}$ such that*

$$k(x, y) = \Phi(x - y) \quad \forall x, y \in \mathbb{R}^d.$$

A kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is radial, iff there exists a $\Phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ such that

$$k(x, y) = \Phi(\|x - y\|_2) \quad \forall x, y \in \mathbb{R}^d.$$

The function Φ is called the radial basis function. Such radial kernels (or radial basis function kernels) can be easily modified by the introduction of a *shape parameter* $\varepsilon > 0$, which modifies the kernel according to

$$k_\varepsilon(x, y) = \Phi(\varepsilon \cdot \|x - y\|_2). \tag{2.1}$$

A lot of properties such as stability of numerical algorithms and error estimates depend on the smoothness of the considered kernel k. For translational invariant kernels, this is usually characterized with help of the decay of the d-dimensional Fourier transform $\hat{\Phi}$ of the function Φ:

- A translational invariant kernel k is called kernel of *finite smoothness* $\tau > d/2$, if there exists constants $c_\Phi, C_\Phi > 0$ such that

$$c_\Phi(1 + \|\omega\|_2^2)^{-\tau} \leq \hat{\Phi}(\omega) \leq C_\Phi(1 + \|\omega\|_2^2)^{-\tau}. \tag{2.2}$$

- A translational invariant kernel k is called kernel of *infinite smoothness*, if for any $k \in \mathbb{N}$ there exists a constant $C_{\Phi,k} > 0$ such that

$$\hat{\Phi}(\omega) \leq C_{\Phi,k}(1 + \|\omega\|_2^2)^{-k}.$$

Well-known examples of strictly positive definite kernels are the Gaussian or the exponential kernel, which are even radial basis function kernels. They are defined as

$$\begin{aligned} k_{\text{Gauss}} &= \exp(-\|x - y\|_2^2), \\ k_{\text{exponential}} &= \exp(-\|x - y\|_2). \end{aligned} \tag{2.3}$$

The exponential kernel is a finitely smooth kernel and the most basic example from the class of Matérn kernels, which provide finitely smooth kernels for any dimension $d \in \mathbb{N}$. The Gaussian kernel on the other hand is an infinitely smooth kernel for any $d \in \mathbb{N}$.
Here, we furthermore introduce the Wendland "$k = 0$" kernel in one dimension because it will be used later on. It is defined as

$$k(x, y) = \min(0, 1 - |x - y|). \tag{2.4}$$

The class of Wendland kernels is of interest because they provide compactly supported kernels.

2.2 Function spaces

Kernels are related to so-called *reproducing kernel Hilbert spaces*. A Hilbert space of real valued functions $f : \Omega \to \mathbb{R}$ is called a reproducing kernel Hilbert space (RKHS), if there exists a *reproducing kernel* k such that:

1. $k(\cdot, x) \in \mathcal{H} \quad \forall x \in \Omega$
2. $f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{H}} \quad \forall x \in \Omega, \forall f \in \mathcal{H}$ (reproducing property)

A well-known theorem from Aronszajn [5, p. 344] (which he attributes to Moore) also shows kind of the reverse direction: Every positive definite kernel k gives rise to a reproducing kernel Hilbert space, which is formalized in the following theorem:

Theorem 2. *Let Ω be a nonempty set and k be a positive definite kernel. Then, there is a unique Hilbert space $\mathcal{H}_k(\Omega)$ of functions on Ω such that k is its reproducing kernel.*

In order to stress the connection to the reproducing kernel k, the notion $\mathcal{H}_k(\Omega)$ was introduced. Depending on the research community, the RKHS of a kernel k is also called the *native space*. The proof of Theorem 2 is quite constructive, and is done by considering the space

$$\mathcal{H}_0 := \operatorname{span}\{k(\cdot, x), x \in \Omega\} \tag{2.5}$$

and showing that it is a pre-Hilbert space with respect to the bilinear product

$$\langle k(\cdot, x), k(\cdot, z)\rangle_{\mathcal{H}_0} := k(x, z).$$

Then, the abstract completion of $\mathcal{H}_0$ with respect to the norm induced by the bilinear product defined before yields the desired RKHS. It remains to show that this abstractly completed space is indeed a space of functions, which follows by showing pointwise convergence due to Cauchy's inequality.

In order to recall the dependence of the RKHS $\mathcal{H}_k(\Omega)$ on the underlying domain Ω, we consider the two sets $\tilde{\Omega} \subset \Omega \subset \mathbb{R}^d$. We define the *restricted kernel* $\tilde{k}$ via

$$\tilde{k} := k_{\tilde{\Omega}} : \tilde{\Omega} \times \tilde{\Omega} \to \mathbb{R}, (x, y) \mapsto k(x, y). \tag{2.6}$$

This mathematical precise distinction will be of importance for Section 3.4. We collect the results [112, Theorem 10.46] and [112, Theorem 10.47].

Theorem 3. *Each function $\tilde{f} \in \mathcal{H}_{\tilde{k}}(\tilde{\Omega})$ has an extension to a function $E\tilde{f} \in \mathcal{H}_k(\Omega)$. Furthermore, $\|Ef\|_{\mathcal{H}_k(\Omega)} = \|f\|_{\mathcal{H}_k(\tilde{\Omega})}$.*

The operator $E : \mathcal{H}_{\tilde{k}}(\tilde{\Omega}) \to \mathcal{H}_k(\Omega)$ is called the *embedding operator*.

Theorem 4. *The restriction $f|_{\tilde{\Omega}}$ of any function $f \in \mathcal{H}_k(\Omega)$ is contained in $\mathcal{H}_{\tilde{k}}(\tilde{\Omega})$ with $\|f|_{\tilde{\Omega}}\|_{\mathcal{H}_{\tilde{k}}(\tilde{\Omega})} \leq \|f\|_{\mathcal{H}_k(\Omega)}$.*

For kernels of finite smoothness, as introduced in Section 2.1, the corresponding RKHS can be characterized with help of Sobolev spaces. We have:

Corollary 5. *Let k be a translation invariant kernel with Fourier transform decay as Eq. (2.2) with $\tau > d/2$. Suppose that Ω has a Lipschitz boundary. Then the RKHS $\mathcal{H}_k(\Omega)$ is norm equivalent to the Sobolev space $H^\tau(\Omega)$, i.e. $\mathcal{H}_k(\Omega) \asymp H^\tau(\Omega)$.*

2.3 Approximation with kernels

Kernels can be used to approximate functions, e.g. by interpolation. For this the standard assumption is $f \in \mathcal{H}_k(\Omega)$, such that the previously introduced results can be easily leveraged.
A starting point for this is provided by the well-known kernel representer theorem [56, 101, 110]. Here we state the version of [105, Theorem 5.5] for the mean squared loss:

Theorem 6. *Consider a positive definite kernel k. Let $X_n = \{x_1, ..., x_n\} \subset \Omega$ be a nonempty set with corresponding target values $Y_n = \{y_1, ..., y_n\}$ and a regularization parameter $\lambda > 0$. Then, there exists a unique solution $s_{n,\lambda}$ satisfying*

$$\frac{1}{n}\sum_{j=1}^{n} |y_j - s_{n,\lambda}(x_j)|^2 + \lambda \|s_{n,\lambda}\|^2_{\mathcal{H}_k(\Omega)} = \inf_{f \in \mathcal{H}_k(\Omega)} \frac{1}{n}\sum_{j=1}^{n} |y_j - f(x_j)|^2 + \lambda \|f\|^2_{\mathcal{H}_k(\Omega)},$$

and it is of the form

$$s_{n,\lambda}(x) = \sum_{j=1}^{n} \alpha_j^{(n)} k(x, x_j), \quad \alpha_j^{(n)} \in \mathbb{R}. \tag{2.7}$$

The vector of coefficients $\alpha^{(n)} = (\alpha_1^{(n)}, ..., \alpha_n^{(n)})^\top \in \mathbb{R}^n$ can be obtained as the solution of the linear equation system

$$(A_{X_n} + \lambda I)\alpha^{(n)} = y, \tag{2.8}$$

where A_{X_n} is the kernel matrix related to X_n and $y = (y_1, ..., y_n)^\top$. The case $\lambda = 0$ usually works only for strictly positive definite kernels. In this case, i.e. exact interpolation, one seeks the interpolant $s_n := s_{n,\lambda} \in \mathcal{H}_k(\Omega)$ of minimal norm $\|s_n\|_{\mathcal{H}_k(\Omega)}$. Then the solution is again given by Eq. (2.7) and the linear equation system from Eq. (2.8) simplifies to

$$A_{X_n}\alpha^{(n)} = y,$$

which is solvable, as the kernel matrix is positive definite.

The case $\lambda > 0$ is usually of interest for (machine learning) applications, where the target data $\{y_1, ..., y_n\}$ is possibly affected by noise. The case $\lambda = 0$ corresponds to exact interpolation of the target values and is useful in case of absence of noise or if it is negligible small, e.g. for approximation theoretical statements.

In the following we provide more information about this interpolation case and assume that the target data $y_j, j = 1, ..., n$ stem from some function $f \in \mathcal{H}_k(\Omega)$, such that the interpolation conditions are

$$s_n(x_j) \equiv s_{n,\lambda=0}(x_j) = f(x_j) \quad \forall j = 1, ..., n. \tag{2.9}$$

Then, it can be shown that the unique interpolant $s_n \in \text{span}\{k(\cdot, x_j), j = 1, ..., n\}$ is equal to the orthogonal projection $\Pi_{V(X_n)}(f)$ of $f \in \mathcal{H}_k(\Omega)$ onto the n-dimensional subspace $V(X_n) := \text{span}\{k(\cdot, x_j), j = 1, ..., n\}$.

A convenient way to estimate the $\|\cdot\|_{L^\infty(\Omega)}$ approximation error between s_n and f is provided by the so-called *power function*, for which several expressions exists:

$$\begin{aligned} P_{X_n}(x) &:= \|k(\cdot,x) - \Pi_{V(X_n)}(k(\cdot,x))\|_{\mathcal{H}_k(\Omega)} && (2.10)\\ &\equiv \sup_{0\neq\tilde{f}\in\mathcal{H}_k(\Omega)} \frac{|(\tilde{f} - \Pi_{V(X_n)}(\tilde{f}))(x)|}{\|\tilde{f}\|_{\mathcal{H}_k(\Omega)}} && (2.11)\\ &\equiv \left(\frac{\det A_{X_n\cup\{x\}}}{\det A_{X_n}}\right)^{1/2}. && (2.12) \end{aligned}$$

We remark that power functions can also be defined for any closed subspace by using Eq. (2.10) and considering the projector onto this closed subspace. From Eq. (2.11) it is obvious that the $\|f - s_n\|_{L^\infty(\Omega)} = \|f - \Pi_{V(X_n)}(f)\|_{L^\infty(\Omega)}$ error can be bounded by the power function. Indeed it holds

$$\begin{aligned} |(f-s_n)(x)| &\leq P_{X_n}(x)\cdot\|f - \Pi_{V(X_n)}(f)\|_{\mathcal{H}_k(\Omega)}\\ &\leq P_{X_n}(x)\cdot\|f\|_{\mathcal{H}_k(\Omega)} \qquad \forall x\in\Omega, \end{aligned} \tag{2.13}$$

and Eq. (2.13) will be called *standard power function estimate*. The estimate $\|f - \Pi_{V(X_n)}(f)\|_{\mathcal{H}_k(\Omega)} \leq \|f\|_{\mathcal{H}_k(\Omega)}$ followed from the best approximation property due to the projection $\Pi_{V(X_n)}$.

The approximation error $\|f - \Pi_{V(X_n)}(f)\|_{L^\infty(\Omega)}$ is frequently bounded with help of the *fill distance* $h_{X_n,\Omega}$, which measures, graphically speaking, the largest hole within Ω without any interpolation point:

$$h_{X_n,\Omega} := \sup_{x\in\Omega}\min_{x_j\in X_n}\|x - x_j\|_2. \tag{2.14}$$

Another quantity of interest is provided by the *separation distance* q_{X_n}, which is usually used to bound instability related quantities, e.g. the smallest eigenvalue of the kernel matrix A_{X_n}:

$$q_{X_n} := \frac{1}{2}\min_{x_i\neq x_j}\|x_i - x_j\|_2.$$

For bounded domains $\Omega\subset\mathbb{R}^d$ it holds

$$\begin{aligned} h_{X_n,\Omega} &\geq c_\Omega\cdot n^{-1/d},\\ q_{X_n} &\leq C_\Omega\cdot n^{-1/d}, \end{aligned} \tag{2.15}$$

which can be derived by volume comparison arguments [75, Section 2.1]. In order to measure the uniformity of points $X_n\subset\Omega$, the ratio $h_{X_n,\Omega}/q_{X_n}$ is considered. A sequence of sets of points $(X_n)_{n\in\mathbb{N}}$ is called *asymptotically uniformly distributed* if there exists a constant $C>0$ such that

$$h_{X_n,\Omega}/q_{X_n} \leq C \qquad \forall n\in\mathbb{N}. \tag{2.16}$$

Somehow imprecise, also a set of points X_n is called asymptotically uniformly distributed, if the sequence of sets $(X_n)_{n\in\mathbb{N}}$ is clear from the context. In the case of asymptotically uniformly distributed points, Eq. (2.15) together with Eq. (2.16) then yields,

$$c_\Omega \cdot n^{-1/d} \leq h_{X_n,\Omega} \leq C \cdot q_{X_n} \leq CC_\Omega \cdot n^{-1/d}. \tag{2.17}$$

In addition to the power function approach, another way to bound the approximation error is provided with help of sampling inequalities. Their basic idea is [87, Section 2]:

> "If a sufficiently smooth function is small on scattered points, and if its derivatives are bounded, it must be small in the whole domain."

For the subsequent purposes it is enough to state the following result, which was taken from [39, Theorem 2.2]

Theorem 7. *Suppose $\Omega \subset \mathbb{R}^d$ is a bounded domain satisfying an interior cone condition and having a Lipschitz boundary. Let $X \subset \Omega$ be a discrete set with sufficiently small fill distance $h = h_{X,\Omega}$. Let $\tau = k + s$ with $k \in \mathbb{N}, 0 \leq s < 1, 1 \leq p < \infty, 1 \leq q \leq \infty, m \in \mathbb{N}_0$ with $k > m + d/p$ if $p > 1$ or $k \geq m + d/p$ if $p = 1$. Then for each $u \in W_p^\tau(\Omega)$ we have that*

$$|u|_{W_q^m(\Omega)} \leq C \left(h^{(\tau - m - d(1/p - 1/q)_+)} \cdot |u|_{W_p^\tau(\Omega)} + h^{-m} \|u|_X\|_\infty \right),$$

where $C > 0$ is a constant independent of u and h and $(x)_+ = \max\{x, 0\}$.

Using $u := f - \Pi_{V(X)}(f)$ gives $u|_X \equiv f - \Pi_{V(X)}(f)|_X = 0$ due to the interpolation conditions. Furthermore, leveraging the norm equivalence of the RKHS of finitely smooth kernels to Sobolev spaces (Corollary 5), allows deriving error bounds for the residual $f - \Pi_{V(X_n)}(f)$ of the kernel interpolant in any Sobolev norm. Using asymptotically uniformly distributed points, these bounds can then also be formulated in terms of the number n of interpolation points X_n by using Eq. (2.17). Due to Eq. (2.11), these bounds then also give bounds on the power function.

The case of regularized approximation in the context of (greedy) kernel interpolation, i.e. $\lambda > 0$ in Theorem 6, is not covered in this thesis in detail. The interested reader is pointed to [95].

2.4 Generalized interpolation

In chapters on generalized interpolation (i.e. here in Section 2.4 and Section 3.6) we will denote the function to be approximated as $u \in \mathcal{H}_k(\Omega)$ instead of $f \in \mathcal{H}_k(\Omega)$.

The use of kernels is not restricted to using inputs $x_j \in \Omega$ and corresponding function values $u(x_j)$ for some $u \in \mathcal{H}_k(\Omega)$ as it was done in Section 2.3. In fact, it is possible to *generalize* the theory presented in Section 2.3 and make use of information from any linear continuous functionals $\lambda_j \in \mathcal{H}_k(\Omega)', j = 1, ..., n$ and seek an approximant $s_n \in \mathcal{H}_k(\Omega)$ that satisfies

$$\lambda_j(s_n) = \lambda_j(u) \quad \forall j = 1, ..., n. \tag{2.18}$$

Table 2.1: Overview on standard and generalized interpolation.

	standard	generalized
Conditions	$X_n \subset \Omega$	$\Lambda_n \subset \mathcal{H}_k(\Omega)'$
Kernel matrix	$(A_{X_n})_{ij} = \langle k(\cdot, x_i), k(\cdot, x_j)\rangle_{\mathcal{H}_k(\Omega)}$	$(A_{\Lambda_n})_{ij} = \langle v_{\lambda_i}, v_{\lambda_j}\rangle_{\mathcal{H}_k(\Omega)}$
Subspace	$V(X_n) = \operatorname{span}\{k(\cdot, x_j), x_j \in X_n\}$	$V(\Lambda_n) = \operatorname{span}\{v_{\lambda_j}, \lambda_j \in \Lambda_n\}$
Power function	$P_{X_n}(x)$	$P_{\Lambda_n}(\lambda)$

As $\lambda_j \in \mathcal{H}_k(\Omega)'$, we can find corresponding Riesz representers which we denote as $v_{\lambda_j} \in \mathcal{H}_k(\Omega)$. We define $V(\Lambda_n) := \{v_{\lambda_j}, \lambda_j \in \Lambda_n\}$. Then it can be shown that the minimum norm solution to Eq. (2.18) is unique and again given by a projection $\Pi_{V(\Lambda_n)}(u)$ with $\Pi_{V(\Lambda_n)} : \mathcal{H}_k(\Omega) \to V(\Lambda_n)$:

$$s_n(x) = \Pi_{V(\Lambda_n)}(f)(x) = \sum_{j=1}^{n} \alpha_j v_{\lambda_j}(x). \tag{2.19}$$

The vector of coefficients $\alpha = (\alpha_1, ..., \alpha_n)^\top$ can be obtained with help of the equation system

$$A_{\Lambda_n}\alpha = (\lambda_j(u))_{j=1}^n \in \mathbb{R}^n,$$

where A_{Λ_n} is the generalized kernel matrix with $(A_{\Lambda_n})_{i,j} = \langle \lambda_i, \lambda_j\rangle_{\mathcal{H}_k(\Omega)}$, $1 \le i, j \le n$.

In order to bound the approximation error, the *generalized power function* is introduced, which can be expressed as

$$P_{\Lambda_n}(\lambda) := \|v_\lambda - \Pi_{V(\Lambda_n)}(v_\lambda)\|_{\mathcal{H}_k(\Omega)} \tag{2.20}$$

$$= \sup_{0 \neq u \in \mathcal{H}_k(\Omega)} \frac{|\lambda(u - \Pi_{V(\Lambda_n)}(u))|}{\|u\|_{\mathcal{H}_k(\Omega)}}, \tag{2.21}$$

and implies the *standard generalized power function estimate*

$$\begin{aligned} |\lambda(u) - \lambda(\Pi_{V(\Lambda_n)}(u))| &\le P_{\Lambda_n}(\lambda) \cdot \|u - \Pi_{V(\Lambda_n)}(u)\|_{\mathcal{H}_k(\Omega)} \\ &\le P_{\Lambda_n}(\lambda) \cdot \|u\|_{\mathcal{H}_k(\Omega)} \qquad \forall u \in \mathcal{H}_k(\Omega). \end{aligned} \tag{2.22}$$

We point out that this gives a bound on the λ-evaluation $\lambda(u - \Pi_{V(\Lambda_n)}(u))$ of the residual. Only for the choice $\lambda \in \mathcal{H}_k(\Omega)'$ such that $v_\lambda = k(\cdot, x)$, we obtain the point evaluation of the residual $u - \Pi_{V(\Lambda_n)}(u)$ itself.

In comparison with Section 2.3 it can be seen that generalized interpolation is indeed a generalization of kernel interpolation: By using $\Lambda_n = \{\delta_{x_j}, x_j \in X_n\}$, the framework of generalized interpolation specializes again to the case of standard interpolation. The corresponding connections are summarized in Table 2.1.

A particularly interesting use case for the framework of generalized interpolation is given by solving PDEs. A linear elliptic PDE with Dirichlet conditions can be written as

$$\begin{aligned} Lu &= f \text{ on } \Omega \ \subset \mathbb{R}^d \\ u &= g \text{ on } \partial\Omega, \end{aligned} \tag{2.23}$$

and the Poisson equation which uses $L := -\Delta$ is a well-known example. By choosing a suitable kernel k such that it holds

$$\begin{aligned} u &\in \mathcal{H}_k(\Omega), \\ \delta_x \circ L &\in \mathcal{H}_k(\Omega)', \quad x \in \Omega, \\ \delta_x &\in \mathcal{H}_k(\Omega)', \quad x \in \partial\Omega, \end{aligned}$$

we can introduce the sets of functionals

$$\begin{aligned} \Lambda_L &:= \{\delta_x \circ L : x \in \Omega\} \\ \Lambda_B &:= \{\delta_x : x \in \partial\Omega\} \\ \Lambda &:= \Lambda_L \cup \Lambda_B. \end{aligned} \tag{2.24}$$

Then the solution $u \in \mathcal{H}_k(\Omega)$ can be approximated in the framework of generalized interpolation by computing the minimum-norm interpolant $\Pi_{V(\Lambda_n)}(u)$ for a suitable chosen set of generalized interpolation conditions $\Lambda_n = \{\lambda_1, ..., \lambda_n\} \subset \Lambda$. The functionals $\lambda \in \Lambda \equiv \Lambda_L \cup \Lambda_B$ from Eq. (2.24) can be conveniently linked to their underlying points $x = x_\lambda$, because $\Omega \cap \partial\Omega = \emptyset$ for open domains Ω. We remark that for this, the solution $u \in \mathcal{H}_k(\Omega)$ does not need to be known, because the generalized interpolation conditions $\lambda_j(s_n) = \lambda_j(u), j = 1, ..., n$ read

$$\begin{cases} (Ls_n)(x_{\lambda_j}) &= f(x_{\lambda_j}) \qquad \lambda_j \in \Lambda_L \\ s_n(x_{\lambda_j}) &= g(x_{\lambda_j}) \qquad \lambda_j \in \Lambda_B \end{cases}$$

i.e. they depend only on the right-hand side functions f, g of the PDE from Eq. (2.23).

2.5 Greedy algorithms

As elaborated in the previous Section 2.3 and Section 2.4, kernel approximants are given by a weighted sum of kernels centered in points $X_n \subset \Omega$. As a good choice of points $X_n \subset \Omega$ can be beneficial in terms of approximation properties or stability concerns, this raises the question how to obtain them. An optimal choice or global search is usually infeasible, therefore *greedy methods* are employed. The basic idea of greedy algorithms is to proceed step-by-step, see [108] for a general overview.
For *standard greedy kernel interpolation algorithms*, this means in particular: We start with an empty set $X_0 = \emptyset$ and then incrementally add points $X_{n+1} := X_n \cup \{x_{n+1}\}$ according to some optimality criterion $\eta^{(n)}$ as

$$x_{n+1} := \arg\max_{x \in \Omega} \eta^{(n)}(x). \tag{2.25}$$

Two remarks are required on Eq. (2.25):

1. For open domains Ω it is not clear, whether a maximum of $\eta^{(n)}(x)$ exists at all, or whether only a supremum exists. To alleviate this technical issue, *weak* selection criteria can be considered, see eg. Definition 18. The analysis is essentially the same.

2. Usually the optimality criterion $\eta^{(n)}$ is zero in the already chosen points, i.e. $\eta^{(n)}(x_j) = 0$ for all $x_j \in X_n$. Therefore, an already chosen point will not be picked again (given $\eta^{(n)}$ is nonnegative), and there is no analytic need to exclude X_n in the search set.

The literature introduces four different selection criteria, namely the P-greedy [22], f-greedy [100] and f/P-greedy [75] and the *psr*-greedy (power-scaled-residual) [31], which we will however call the $f \cdot P$-greedy because it fits better into the notation introduced later on. Their selection criteria are given as

i. P-greedy: $\eta_P^{(n)}(x) = P_n(x)$,

ii. f-greedy: $\eta_f^{(n)}(x) = |r_n(x)|$,

iii. f/P-greedy: $\eta_{f/P}^{(n)}(x) = |r_n(x)|/P_n(x)$,

iv. $f \cdot P$-greedy: $\eta_{f\cdot P}^{(n)}(x) = |r_n(x)| \cdot P_n(x)$,

whereby $r_n := f - s_n$ denotes the residual. We remark that the f/P-greedy algorithm is not necessarily well-defined, see the counter example in [119, Example 6]. This further motivates considering weak selection criteria, see e.g. Definition 18. It is immediate that these different selection criteria imply different properties on the resulting interpolant s_n, especially on the convergence rate $\|f - s_n\| \to 0$ measured in some suitable norm, e.g. $\|\cdot\| = \|\cdot\|_{L^\infty(\Omega)}$. Results on such convergence rates are discussed in detail in Section 3.1. For efficient numerical implementations of these greedy algorithms, but also for analytic results, the introduction of a *Newton basis* is helpful and was done in [75]. This Newton basis $\{v_j\}_{j=1}^n$ (not to be confused with the Riesz representers $\{v_{\lambda_j}\}_{j=1}^n$ from the generalized interpolation in Section 2.4) can be obtained by a Gram Schmidt orthogonalization of the subspace $V(X_n) = \operatorname{span}\{k(\cdot, x_j), x_j \in X_n\}$ with respect to $\langle\cdot,\cdot\rangle_{\mathcal{H}_k(\Omega)}$. Using the Newton basis, the kernel interpolant from Eq. (2.7) can be written as

$$s_n(x) = \sum_{j=1}^{n} \langle f, v_j\rangle_{\mathcal{H}_k(\Omega)} v_j(x). \tag{2.26}$$

A crucial advantage is, that, in order to update s_n to s_{n+1}, only the latest element $v_{n+1}(x)$ with corresponding coefficient $\langle f, v_{n+1}\rangle_{\mathcal{H}_k(\Omega)}$ needs to be added, and there is no need to recompute all the coefficients $(\alpha_j^{(n)})_{j=1}^n$ as in Eq. (2.7). Furthermore, it holds

$$\langle f, v_j\rangle_{\mathcal{H}_k(\Omega)} = \frac{|(f - s_{j-1})(x_j)}{P_{X_{j-1}}(x_j)}, \qquad j = 1, 2, 3, ... \tag{2.27}$$

If $s_n \overset{n\to\infty}{\longrightarrow} f$ in $\mathcal{H}_k(\Omega)$, then it follows that

$$\sum_{j=0}^{\infty} \left(\frac{(f - s_j)(x_{j+1})}{P_{X_j}(x_{j+1})} \right)^2 = \|f\|^2_{\mathcal{H}_k(\Omega)}, \tag{2.28}$$

and in general we have always the inequality, i.e. "$\leq$" in Eq. (2.28).

Greedy algorithms can also be formulated for generalized interpolation, and also a Newton basis with corresponding formulas can be derived: Following Section 2.4, given a set $\Lambda \subset \mathcal{H}_k(\Omega)'$ of generalized interpolation conditions, we start with an empty set $\Lambda_0 := \emptyset$

and incrementally add functionals $\Lambda_{n+1} := \Lambda_n \cup \{\lambda_{n+1}\}$ according to some optimality criterion $\eta^{(n)}$ as

$$\lambda_{n+1} := \arg\max_{\lambda \in \Lambda} \eta^{(n)}(\lambda), \tag{2.29}$$

which is a straightforward generalization of Eq. (2.25). The construction of a Newton basis works again by applying the Gram Schmidt orthogonalization procedure with respect to $\langle \cdot, \cdot \rangle_{\mathcal{H}_k(\Omega)}$ to the Riesz representer of the chosen functionals, i.e. $\{v_{\lambda_1}, ..., v_{\lambda_n}\}$. As $V(\Lambda_n) \equiv \text{span}\{v_{\lambda_1}, ..., v_{\lambda_n}\} = \text{span}\{v_1, ..., v_n\}$ it follows that Eq. (2.26) still holds, because the interpolant s_n is given as the projection onto $V(\Lambda_n)$, see Eq. (2.19).
The formula for calculating the Newton coefficients is modified to

$$\langle u, v_j \rangle_{\mathcal{H}_k(\Omega)} = \frac{\lambda_j(u - s_{j-1})}{P_{\Lambda_{j-1}}(\lambda_j)}, \qquad j = 1, 2, 3, ... \tag{2.30}$$

and Eq. (2.28) turns into

$$\sum_{j=0}^{\infty} \left(\frac{|\lambda_{j+1}(u - s_j)|}{P_{\Lambda_j}(\lambda_{j+1})} \right)^2 = \|u\|^2_{\mathcal{H}_k(\Omega)}, \tag{2.31}$$

for generalized interpolation if $s_n \overset{n\to\infty}{\longrightarrow} u$ in $\mathcal{H}_k(\Omega)$. The familiar reader observes again, that Eq. (2.31) simplifies to Eq. (2.28) and Eq. (2.30) simplifies to Eq. (2.27) for $\lambda_j = \delta_{x_j}, j = 1, ..., n$.

For solving PDEs by greedy kernel approximation, i.e. using the choice Eq. (2.24) for the sets of functionals, [99] introduced the choice

$$\lambda_{n+1} := \arg\max_{\lambda \in \Lambda \setminus \Lambda_n} P_{\Lambda_n}(\lambda),$$

which can be seen as a *generalized P-greedy* selection criterion.

2.6 Deep models

Deep learning [42] describes a family of machine learning methods, which make use of a multilayer model structure that allows to express the model as a concatenation of simple functions. The most prominent examples are neural networks. Due to their multilayer structure, these deep models allow for representation or feature learning: The first layers can extract features, which are then used for the subsequent learning tasks. This renders such deep models suitable for high dimensional datasets occurring in applications, as they can automatically detect a suitable lower dimensional representation of the data. Another benefit is their natural scaling to large data sets, because they do not rely (in contrast to flat kernel methods) on kernel evaluations of pairs of data points which scales quadratically in the number of samples.
These two points – no feature learning and disadvantageous scaling in the data set size – pose serious limitations to standard kernel methods. Therefore, it is imminent to ask for combinations of kernel methods and deep models in order to address these challenges, for which two approaches are presented in Chapter 4. The following Section 2.6.1 and Section 2.6.2 thus provide more background information on neural networks and deep kernels.

2.6.1 Neural networks

In contrast to standard kernel methods as described in the previous sections, neural networks (NNs) are an instance of deep models and parametric models [42]. In their most basic form, i.e. feedforward neural networks, the mapping f can be written as a concatenation of basic functions:

$$f_{\mathrm{NN}}(x) = f_L \circ ... \circ f_1(x). \tag{2.32}$$

Every mapping f_j, $j = 1, ..., L$ describes one of the L layers and they are of the form

$$\begin{aligned} f_l(x) &= \sigma(W_l x + b_l), \qquad 1 \leq l \leq L-1, \\ f_L(x) &= W_L x + b_L, \end{aligned} \tag{2.33}$$

with dimensionwise acting activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and weight matrix $W_l \in \mathbb{R}^{d_l \times d_{l-1}}$ and bias vector $b_l \in \mathbb{R}^{d_l}$. Due to these parameters $\Theta := \{W_l\}_{l=1}^L \cup \{b_l\}_{l=1}^L$, neural networks are an instance of parametric models.
Given some input data $X_n := \{x_1, ..., x_n\} \subset \mathbb{R}^d$ with corresponding output values $Y_n := \{y_1, ..., y_n\} \subset \mathbb{R}$, the optimization of such a neural network $f_{\mathrm{NN},\Theta} \equiv f_{\mathrm{NN}}$ refers to a modification of the parameters Θ to minimize the cost functional

$$J(\Theta, X_n, Y_n) \equiv \sum_{i=1}^{n} \mathcal{L}(y_i, f_{\mathrm{NN},\Theta}(x_i)) + \mathcal{R}(\Theta), \tag{2.34}$$

which consists of a loss part $\mathcal{L}$ and a regularization part $\mathcal{R}$. Frequent choices are the least squares loss and a weight penalization via

$$\begin{aligned} \mathcal{L}(y, f_{\mathrm{NN},\Theta}(x)) &:= \frac{1}{2}|y - f_{\mathrm{NN},\Theta}(x)|^2 \\ \mathcal{R}(\Theta) &:= \sum_{i=1}^{L} \lambda_{i,W} \|W_l\|^2 + \sum_{i=1}^{L} \lambda_{i,b} \|b_l\|^2 \end{aligned}$$

for regularization parameters $\lambda_{i,W}, \lambda_{i,b} \geq 0$, $i = 1, ..., L$. The parameters Θ are usually optimized using a gradient descent mini-batch algorithm, whereby the Adam optimizer [57] is a popular choice.
Neural networks are known to be universal approximators under mild conditions, which means that continuous functions can be approximated arbitrarily well: The case of wide networks (i.e. $d_1, ..., d_{L-1} \to \infty$) was analyzed e.g. in [52, 64, 85] and for the case of deep networks (i.e. $L \to \infty$) see e.g. [32, 49, 107]. For the case of deep networks, much focus was put on the popular ReLU activation function $\sigma(x) = \max(0, x)$.
There are also some connections to kernel methods, which are partly of interest for the following: First, there are representer theorems for neural networks, see e.g. [82, 109]. Connections between these representer theorems and the kernel representer theorems of Theorem 6 and Theorem 8 are discussed in Section 4.2.5. In order to emphasize the feature learning ability of neural networks, Eq. (2.32) and Eq. (2.33) can be written as

$$f_{\mathrm{NN}}(x) = W_L \left(f_{L-1} \circ ... \circ f_1(x)\right) + b_L,$$

which highlights that a neural network is essentially a linear model in the feature $f_{L-1} \circ ... \circ f_1(x)$ of x. As the mappings $f_1, ..., f_{L-1}$ are described by the parameters $\{W_l\}_{l=1}^{L-1}$ and $\{b_l\}_{l=1}^{L-1}$ which can be optimized, neural networks allow for feature learning. A particular interesting case are convolutional neural networks, where the learned features even have a visual interpretation and explanation [43]. This can be contrasted with kernel models, as the kernel model from Eq. (2.19) reads

$$s_{n,\lambda}(x) = \sum_{j=1}^{n} \alpha_j^{(n)} k(x, x_j)$$

using a fixed preselected kernel k, that does not adapt to the data (beyond the choice of e.g. a shape parameter as in Eq. (2.1)). Chapter 4 starts at this point and proposes kernel models which adapt to the data just as neural networks do, but keeping benefits of kernel methods.
Another interesting connection between neural networks and kernels was initially described in [16] which introduced arc-cosine kernels, that can be seen as infinite width limit of one hidden layer neural networks. This idea – obtaining kernels from the infinite width limit of neural networks – was followed up in [53] where the *neural tangent kernel* (NTK) was introduced, which is defined by the feature map obtained by the gradients of the neural network. As the construction defined the kernel recursively, it thus also works for multilayer neural networks and especially using quite general activation functions. The crucial insight was, that the neural tangent kernel becomes deterministic in the limit of infinite width, and that then the training behavior of such infinite width neural networks can be described exactly with help of the stationary NTK. This connection allowed to analyze the generalization performance of neural networks. This work has sparked a lot of research recently, see e.g. [6, 11, 50, 63, 131].

2.6.2 Data adapted and deep kernel models

As argued in Section 1.1 the choice of a proper kernel is desired in order to obtain a good kernel model. Therefore, it makes sense to strive for data adapted kernels, which take into account properties of the data set or the underlying problem, even before computing a model. In the context of approximation theory and in particular function approximation, as dealt in Chapter 3, it can be argued [19] that the optimal kernel for the approximation of a function $f : \Omega \to \mathbb{R}$ is given by

$$k(x, z) = f(x)f(z),$$

because it allows to perfectly recover a function with only one sample point x_1 (that satisfies $f(x_1) \neq 0$) as

$$f(x) = \frac{1}{f(x_1)} k(x, x_1).$$

However – especially if only partial information by data points is provided – it is hardly possible to find this optimal kernel.

Nevertheless, it is still possible to obtain a better suited data adapted kernel, though it might not be the optimal kernel. Due to the success of deep models especially via neural

networks, it is imminent to try to obtain data adapted kernels by using a multilayered kernel, i.e. a *deep kernel*. A theoretical milestone for the mathematical foundation of such a deep kernel was set in a deep kernel representer theorem in [13], where a generalization of the standard kernel representer theorem from Theorem 6 was given. A predecessor of this statement for two layers was presented in [24, Section 3.1]. The more general deep kernel representer theorem of [13] is restated in the following (with notation adopted to this thesis) for convenience: For this, a set of corresponding input-output values $(x_i, y_i) \in \mathbb{R}^{d_{\text{in}}+d_{\text{out}}}$, $i = 1, ..., N$ is considered, which is to be approximated with a multilayered function $h = f_L \circ .. \circ f_1$:

Theorem 8. *Let $\mathcal{H}_1, .., \mathcal{H}_L$ be reproducing kernel Hilbert spaces of functions with finite-dimensional domains D_l and ranges $R_l \subseteq \mathbb{R}^{d_l}$ with $d_l \in \mathbb{N}$ for $l = 1, .., L$ such that $R_l \subseteq D_{l+1}$ for $l = 1, .., L-1, D_1 = \Omega$ and $R_L \subset \mathbb{R}$. Let furthermore $\mathcal{L} : \mathbb{R}^2 \to [0, \infty]$ be an arbitrary loss function and let $\Theta_1, .., \Theta_L : [0, \infty) \to [0, \infty)$ be strictly monotonically increasing functions. Then, a set of minimizers $(f_l^*)_{l=1}^L$ with $f_l^* \in \mathcal{H}_l$ of*

$$J(f_1, .., f_L) := \sum_{i=1}^{N} \mathcal{L}(y_i, f_L \circ .. \circ f_1(x_i)) + \sum_{l=1}^{L} \Theta_l(\|f_l\|_{\mathcal{H}_l^2})$$

fulfills $f_l^* \in \tilde{V}_l \subset \mathcal{H}_l$ for all $l = 1, .., L$ with

$$\tilde{V}_l := \text{span}\{K_l(\ \cdot\ , f_{l-1}^* \circ .. \circ f_1^*(x_i))e_{k_l} | i = 1, .., N, k_l = 1, .., d_l\},$$

where K_l denotes the reproducing kernel of $\mathcal{H}_l$ and $e_{k_l} \in \mathbb{R}^{d_l}$ is the k_l-th unit vector.

It is remarked that the cost functional J from Theorem 8 is very similar to the cost term used for the optimization of neural networks, see Eq. (2.34). Overall Theorem 8 gives rise to a deep kernel model s_{deep} as

$$s_{\text{deep}}(\cdot) = \sum_{i=1}^{N} \mathcal{K}_L(\cdot, x_i)\alpha_i \tag{2.35}$$

using the deep kernel

$$\mathcal{K}_L(x, z) = K_L(f_{L-1} \circ .. \circ f_1(x), f_{L-1} \circ .. \circ f_1(z)). \tag{2.36}$$

Chapter 3

Kernel approximation

In contrast to the next Chapter 4, this chapter mainly deals with the analysis of already existing methods and algorithms and by doing so solves some open problems. Moreover, the results and introduced analysis techniques also ease the way to derive new methods. In order to keep the outline in this section as abstract as possible, we try to mainly rely on reproducing kernel Hilbert space arguments most of the time and only make use of arguments from e.g. the theory of Sobolev spaces when necessary.

Section 3.1 starts by summarizing the literature and current state of the art, and Section 3.2 proceeds by introducing required special tools from the literature, which will be leveraged in the forthcoming analysis. Section 3.3 proceeds by generalizing and extending those tools, which will be then used in the subsequent sections: Section 3.4 leverages greedy algorithms as a tool to derive results on general kernel interpolation. Section 3.5 then actually analyzes greedy algorithms itself, and Section 3.6 briefly discusses generalized greedy algorithms. Note that there is a change of the order of presentation here in comparison with the chronological publication order of the papers, which was done in order to highlight the pipeline of the results. Section 3.7 concludes by discussing the results, commenting on them and providing an outlook.

3.1 Literature and state of the art

After the respective introduction of the different greedy kernel algorithms (see Section 2.5 for more details), their analysis was quite tough: After [22] defined the P-greedy algorithm, the provided convergence analysis derived a convergence rate as $\|P_n\|_{L^\infty(\Omega)} \leq n^{-1/d}$ [22, Theorem 4.3.] based on the use of the mean value theorem. Then, [75] defined the f-greedy algorithm and provided a decay rate as $\min_{j=1,\dots,n} \|f - s_{f,X_n}\|_{L^\infty(\Omega)} \leq Cn^{-1/d}$ [75, Korollar 3.3.8]. Afterward, [126] introduced and analyzed the f/P-greedy algorithm, but the proven decay rate holds only for a very small set of functions [126, Theorem 2.8].

Based on a more general setting than kernel interpolation, [23] and the predecessor paper [12] analyzed performance of greedy algorithms in general Banach and Hilbert spaces: It was shown that their considered greedy algorithms essentially realize the same convergence behavior as the corresponding Kolmogorov widths of the considered problem. This analysis was first used by [92] to connect these results to greedy kernel approximation, which established a strong tool. We remark that the paper [37] established indeed the

same connection two years later, probably unaware of [92]. Essentially the considered Kolmogorov width of [23] turned out to be upper boundable by a sampling number of the reproducing kernel Hilbert space, and thus the P-greedy algorithm essentially realizes the same kind of convergence as the corresponding sampling numbers of the considered problem. This connection will also be essential for the forthcoming analysis, and therefore it will be explained in more depth in Section 3.2. Despite this progress in the analysis of the target data-independent P-greedy algorithm, a powerful analysis of target data-dependent algorithms still remained an open problem till the work presented in Section 3.5 and published in [122].

Greedy algorithms can not only be investigated for interpolation or regularized interpolation, but also for generalized interpolation and therefore especially for solving PDEs, see Section 2.4. An initial analysis of a P-greedy like algorithm for symmetric kernel collocation was provided in [99], which generalized the connection established in [92] to generalized interpolation problems, see [99, Section 4]. Section 3.6 comments on generalizing the analysis from Section 3.5 to target data-dependent generalized greedy kernel algorithms.

Much of the analysis on error estimates for kernel approximation focussed on uniformly distributed points and was thus measured in terms of a (local) fill distance, see e.g. [77, 97, 129]. Estimates based on fill distances can be transformed to statements in the number of points for asymptotically distributed points by using Eq. (2.17). However, when working with fill distances, assumptions on the domain like Lipschitz boundary are frequently imposed. These restrictive assumptions on the domain are not required when working with local fill distances [129], nevertheless, these local distances require a clustering of points around cusps and thus do not allow converting them into estimates based on the number of interpolation points for non-Lipschitz domains. This will be removed by the technique introduced in Section 3.4, which directly works in the number of interpolation points and thus circumvents fill distances.

3.2 Background

The following two subsections distinguish between an *abstract setting* and a *kernel setting*, which are both required for the forthcoming analysis of greedy kernel algorithms. This distinction is done to separate the more abstract analysis which takes part in arbitrary Hilbert spaces, and the connection which allows transferring this connection to greedy kernel interpolation.

3.2.1 Abstract setting

A key tool for the analysis of greedy kernel algorithms was provided in [92], which made use of results from [23]. We start by introducing the required terminology and results of [23] and subsequently showing how [92] connected them to kernel approximation:

The paper [23] considers a Hilbert space $\mathcal{H}$ and a compact subset $\mathcal{F}$ which is assumed to satisfy $\|f\| \leq 1$ for all $f \in \mathcal{F}$. We assume $\mathcal{F}$ to contain infinitely many elements, otherwise the forthcoming analysis is trivial. The compactness of $\mathcal{F}$ ensures that $\mathcal{F}$ can be well approximated with finite dimensional subspaces. Then a sequence $\{f_0, f_1, ...\}$ of

elements from $\mathcal{F}$ is constructed by a greedy algorithm, which starts with any $f_0 \in \mathcal{F}$ that satisfies

$$\|f_0\| \geq \gamma \cdot \max_{f \in \mathcal{F}} \|f\|$$

for a preselected $\gamma \in (0, 1]$. Then, defining $V_n := \text{span}\{f_0, ..., f_{n-1}\}$, the next element $f_n \in \mathcal{F}$ is chosen such that it satisfies

$$\text{dist}(f_n, V_n)_{\mathcal{H}} \geq \gamma \cdot \sup_{f \in \mathcal{F}} \text{dist}(f, V_n)_{\mathcal{H}}. \tag{3.1}$$

This algorithm is called *weak greedy algorithm with constant* γ. As $\mathcal{H}$ is a Hilbert space, the distances can be computed by the distance to the best approximating element, which is given by the projection, i.e. it holds $\text{dist}(f, V_n) = \|f - \Pi_{V_n}(f)\|_{\mathcal{H}}$ and $\Pi_{V_n} : \mathcal{H} \to V_n$ is the orthogonal projector onto the n-dimensional subspace $V_n \subset \mathcal{H}$. Especially for $\gamma < 1$, but also for $\gamma = 1$, the resulting sequence does not need to be unique. Nevertheless, the subsequent statements hold for any sequence which is constructed in this way.

For the analysis of the approximation properties of the subspace V_n generated by $f_0, ..., f_{n-1}$, the following quantities are defined:

$$\begin{aligned} \sigma_n(\mathcal{F})_{\mathcal{H}} &:= \sup_{f \in \mathcal{F}} \text{dist}(f, V_n)_{\mathcal{H}}, \\ d_n(\mathcal{F})_{\mathcal{H}} &:= \inf_{Y_n \subset \mathcal{H}} \sup_{f \in \mathcal{F}} \text{dist}(f, Y_n)_{\mathcal{H}}, \end{aligned} \tag{3.2}$$

where the infimum $\inf_{Y_n \subset \mathcal{H}}$ runs over all n-dimensional subspaces Y_n of $\mathcal{H}$.

The first quantity $\sigma_n(\mathcal{F})_{\mathcal{H}}$ measures how well the n-dimensional subspace V_n approximates $\mathcal{F}$, while the second quantity $d_n(\mathcal{F})_{\mathcal{H}}$ measures how well an optimal n-dimensional subspace approximates $\mathcal{F}$. The quantity $d_n(\mathcal{F})_{\mathcal{H}}$ is also called the *Kolmogorov n-width* of $\mathcal{F}$ in $\mathcal{H}$. In the following we frequently drop the index $\mathcal{H}$ and the argument $\mathcal{F}$, if they are clear from the context.

Given those definitions, we recall the technical statement [23, Lemma 2.1] as Lemma 9 and subsequently state [23, Theorem 3.2] as Theorem 10, which will be of importance to us:

Lemma 9. *Let $G = (g_{i,j})$ be a $K \times K$ lower triangular matrix with rows $\boldsymbol{g}_1, ..., \boldsymbol{g}_K$, W be any m-dimensional subspace of $\mathbb{R}^K$, and P be the orthogonal projection of $\mathbb{R}^K$ onto W. Then*

$$\prod_{i=1}^{K} g_{i,i}^2 \leq \left\{ \frac{1}{m} \sum_{i=1}^{K} \|P\boldsymbol{g}_i\|_2^2 \right\}^m \left\{ \frac{1}{K-m} \sum_{i=1}^{K} \|\boldsymbol{g}_i - P\boldsymbol{g}_i\|_2^2 \right\}^{K-m}$$

where $\|\cdot\|_2$ is the Euclidean norm of a vector in $\mathbb{R}^K$.

Theorem 10. *For the weak greedy algorithm with constant γ in a Hilbert space $\mathcal{H}$ and for any compact set $\mathcal{F}$, we have the following inequalities between $\sigma_n := \sigma_n(\mathcal{F})_{\mathcal{H}}$ and $d_n := d_n(\mathcal{F})_{\mathcal{H}}$, for any $N \geq 0, K \geq 1$, and $1 \leq m < K$:*

$$\prod_{i=1}^{K} \sigma_{N+i}^2 \leq \gamma^{-2K} \left(\frac{K}{m} \right)^m \left(\frac{K}{K-m} \right)^{K-m} \sigma_{N+1}^{2m} d_m^{2K-2m}.$$

The proof of Theorem 10 is based on the application of the technical Lemma 9. We do not restate it here, because later on a related proof is given for a modified statement. A conclusion of Theorem 10 is provided by the following corollary, which is [23, Corollary 3.3]:

Corollary 11. *For the weak greedy algorithm with constant γ in a Hilbert space $\mathcal{H}$, we have the following:*

1. *For any compact set $\mathcal{F}$ and $n \geq 1$, we have*
$$\sigma_n(\mathcal{F}) \leq \sqrt{2}\gamma^{-1} \min_{1 \leq m < n} d_m^{\frac{n-m}{n}}(\mathcal{F})$$
2. *If $d_n(\mathcal{F}) \leq C_0 n^{-\alpha}$, $n = 1, 2, ...,$ then $\sigma_n(\mathcal{F}) \leq C_1 n^{-\alpha}$, $n = 1, 2, ...,$ with $C_1 := 2^{5\alpha+1}\gamma^{-2}C_0$.*
3. *If $d_n(\mathcal{F}) \leq C_0 e^{-c_0 n^\alpha}$, $n = 1, 2, ...,$ then $\sigma_n(\mathcal{F}) \leq \sqrt{2C_0}\gamma^{-1} e^{-c_1 n^\alpha}$, $n = 1, 2, ...,$ where $c_1 = 2^{-1-2\alpha} c_0$.*

3.2.2 Kernel setting

In the following, we show how [92] used the abstract setting from [23] in order to derive convergence rates for the P-greedy algorithm: In view of Section 3.2.1, it was chosen $\mathcal{H} = \mathcal{H}_k(\Omega)$ and $\mathcal{F} = \{k(\cdot, x), x \in \Omega\}$. This choice allows identifying any $f \in \mathcal{F}$ uniquely with an $x \in \Omega$. For the subspace $V_n \subset \mathcal{H}$ this means

$$V_n \equiv \operatorname{span}\{f_0, \dots, f_{n-1}\} = \operatorname{span}\{k(\cdot, x_i) \mid i = 1, \dots, n\} \equiv V(X_n). \tag{3.3}$$

With these choices, which are also summarized in Table 3.1, it is possible to express the quantities $\sigma_n(\mathcal{F}), d_n(\mathcal{F})$ from Eq. (3.2) as

$$\begin{aligned} \sigma_n(\mathcal{F}) &\equiv \sup_{f \in \mathcal{F}} \operatorname{dist}(f, V_n)_{\mathcal{H}} = \sup_{f \in \mathcal{F}} \|f - \Pi_{V_n}(f)\|_{\mathcal{H}} \\ &= \sup_{x \in \Omega} \|k(\cdot, x) - \Pi_{V(X_n)}(k(\cdot, x))\|_{\mathcal{H}_k(\Omega)} = \|P_{X_n}\|_{L^\infty(\Omega)}, \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} d_n(\mathcal{F}) &\equiv \inf_{Y_n \subset \mathcal{H}} \sup_{f \in \mathcal{F}} \operatorname{dist}(f, Y_n)_{\mathcal{H}} = \inf_{Y_n \subset \mathcal{H}} \sup_{f \in \mathcal{F}} \|f - \Pi_{Y_n}(f)\|_{\mathcal{H}} \\ &\leq \inf_{Y_n \subset \operatorname{span} \mathcal{F}} \sup_{f \in \mathcal{F}} \|f - \Pi_{Y_n}(f)\|_{\mathcal{H}_k(\Omega)} \leq \inf_{X_n \subset \Omega} \sup_{x \in \Omega} P_{X_n}(x) \\ &= \inf_{X_n \subset \Omega} \|P_{X_n}\|_{L^\infty(\Omega)}, \end{aligned} \tag{3.5}$$

We point out, that due to the choice of $\mathcal{F}$, the Kolmogorov width $d_n(\mathcal{F})$ could be bounded from above by a sampling number $\inf_{X_n \subset \Omega} \|P_{X_n}\|_{L^\infty(\Omega)}$.
Using the unique identification $f \in \mathcal{F} \Leftrightarrow x \in \Omega$, $\mathcal{F} \ni f_n \Leftrightarrow x_{n+1} \in \Omega$, the greedy selection criterion of Eq. (3.1) can be expressed as

$$\begin{aligned} &\Leftrightarrow && \operatorname{dist}(k(\cdot, x_{n+1}), V(X_n)) \geq \gamma \cdot \sup_{x \in \Omega} \operatorname{dist}(k(\cdot, x), V(X_n)) \\ &\Leftrightarrow && \|k(\cdot, x_{n+1}) - \Pi_{V(X_n)}(k(\cdot, x_{n+1}))\|_{\mathcal{H}_k(\Omega)} \geq \gamma \cdot \sup_{x \in \Omega} \|k(\cdot, x) - \Pi_{V(X_n)}(k(\cdot, x))\|_{\mathcal{H}_k(\Omega)} \\ &\Leftrightarrow && P_{X_n}(x_{n+1}) \geq \gamma \cdot \|P_{X_n}\|_{L^\infty(\Omega)}. \end{aligned}$$

In view of Section 2.5, one can see that the greedy selection criterion of the abstract setting is exactly the (weak) P-greedy selection criterion.

In order to be able to leverage Theorem 10 or Corollary 11, one needs to check the remaining requirements:

1. First, we note that the basic assumption $\|f\| \leq 1$ for all $f \in \mathcal{F}$ is equivalent to

$$\sup_{x\in\Omega} \|k(\cdot,x)\|_{\mathcal{H}_k(\Omega)} = \sup_{x\in\Omega} \sqrt{k(x,x)} < 1, \tag{3.6}$$

   whereby the last inequality can be ensured easily for e.g. translational invariant kernels after suitable scaling.

2. Second, we need to check that $\mathcal{F}$ is indeed a compact subset of $\mathcal{H}_k(\Omega)$. This was done in [92] and [122] by leveraging [86, Prop. 1.2 (page 10)], which states that a set $\mathcal{F}$ is compact iff $d_n(\mathcal{F}) \to 0$ and $\mathcal{F}$ is bounded: The boundedness of $\mathcal{F}$ indeed follows from Eq. (3.6) and $d_n(\mathcal{F}) \to 0$ holds as soon as there exists a sequence of sets $(X_n)_{n\in\mathbb{N}}$ such that $\inf_{X_n\subset\Omega} \|P_{X_n}\|_{L^\infty(\Omega)} \overset{n\to\infty}{\longrightarrow} 0$, which frequently holds, e.g. if $h_{X_n,\Omega} \overset{n\to\infty}{\longrightarrow} 0$.
   However, this argumentation via [86, Prop. 1.2 (page 10)] is indeed not complete, as [86] assumes the considered sets to be closed in the beginning of [86, Chapter II]. We discuss this issue in more detail in Section 3.3.1.

With this choice of $\mathcal{H} = \mathcal{H}_k(\Omega)$ and $\mathcal{F} = \{k(\cdot,x), x \in \Omega\}$ it is possible to use Theorem 10 or in particular Corollary 11: Using Eq. (3.5), any asymptotic rate for $\inf_{X_n\subset\Omega} \|P_{X_n}\|_{L^\infty(\Omega)}$ (e.g. a rate realized by well distributed points $X_n \subset \Omega$) carries over to a rate on the decay of the power function for the P-greedy algorithm. This is essentially summarized in [92, Theorem 4.1].

Table 3.1: Connection between the abstract setting and the kernel setting.

Abstract setting	$\mathcal{H}$	$f \in \mathcal{F}$	$V_n \equiv \operatorname{span}\{f_0,\dots,f_{n-1}\}$
Kernel setting	$\mathcal{H}_k(\Omega)$	$k(\cdot,x),\ x\in\Omega$	$V(X_n) \equiv \operatorname{span}\{k(\cdot,x_i), x_i \in X_n\}$

3.3 Analysis of greedy algorithms in an abstract setting

We extend the analysis of [23], which was presented in Section 3.2 in two ways: First, in Section 3.3.1, we elaborate why it is possible to generalize the assumption on $\mathcal{F}$ being a compact set to $\mathcal{F}$ being a precompact set. Second, in Section 3.3.2, we derive more general results than presented in [23] within the same framework. Especially, we formulate these statements using $\mathcal{F} \subset \mathcal{H}$ precompact.

3.3.1 Precompact sets $\mathcal{F} \subset \mathcal{H}$

In Section 3.2.2 we already discussed that the choice of $\mathcal{F} \subset \mathcal{H}$ via $\mathcal{F} := \{k(\cdot, x), x \in \Omega\}$ does not necessarily yield a compact set: If $\Omega \subset \mathbb{R}^d$ is not a closed set, then also $\mathcal{F}$ is not closed and thus in particular not compact. We have the following Corollary 12:

Corollary 12. *Let $\Omega \subset \mathbb{R}^d$ be a bounded set and let $k : \overline{\Omega} \times \overline{\Omega} \to \mathbb{R}$ be a strictly positive definite continuous kernel. Then it holds*

$$\overline{\{k(\cdot, x), x \in \Omega\}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}} = \{k(\cdot, x), x \in \overline{\Omega}\}$$

as sets within $\mathcal{H}_k(\Omega)$.

We remark that $k(\cdot, x) : \Omega \to \mathbb{R} \in \mathcal{H}_k(\Omega)$ for $x \in \partial\Omega$ follows from Theorem 4.

Proof. We define $\mathcal{F} := \{k(\cdot, x), x \in \Omega\}$ and are interested in $\overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}$:

"$\subseteq$" Let $f \in \overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}$, i.e. there exists a sequence $(f_n)_{n\in\mathbb{N}} \subset \mathcal{F}$ such that $\|f - f_n\|_{\mathcal{H}_k(\Omega)} \to 0$, in particular $(f_n)_{n\in\mathbb{N}}$ is a Cauchy sequence. We use the unique identification $f_n \leftrightarrow k(\cdot, x_n)$ (because k is strictly positive definite) to obtain a sequence $(x_n)_{n\in\mathbb{N}} \subset \Omega$. As $\overline{\Omega}$ is compact, there exists an accumulation point $x^* \in \overline{\Omega}$ such that for a subsequence $(n_j)_{j\in\mathbb{N}} \subset \mathbb{N}$ it holds $x_{n_j} \overset{j\to\infty}{\longrightarrow} x^*$. We show now that $f = k(\cdot, x^*)$, and for this it is sufficient to show that $f(x) = k(x, x^*)$ for all $x \in \Omega$:

$$\begin{aligned} |k(x, x^*) - f(x)| &= |k(x, x^*) - k(x, x_{n_j}) + k(x, x_{n_j}) - f(x)| \\ &\leq |k(x, x^*) - k(x, x_{n_j})| + |k(x, x_{n_j}) - f(x)| \\ &\leq |k(x, x^*) - k(x, x_{n_j})| + |f_{n_j}(x) - f(x)|. \end{aligned}$$

The first summand vanishes due to the continuity of k, and the second summand vanishes due to $|f_{n_j}(x) - f(x)| \leq \|k(\cdot, x)\|_{\mathcal{H}_k(\Omega)} \cdot \|f_{n_j} - f\|_{\mathcal{H}_k(\Omega)}$. As $x^* \in \overline{\Omega}$, we have $f \in \{k(\cdot, x), x \in \overline{\Omega}\}$.

"$\supseteq$" Consider $k(\cdot, x^*)$ with $x^* \in \overline{\Omega}$. If $x^* \in \Omega$, then the inclusion is trivial. For $x^* \in \partial\Omega$, consider any sequence $(x_n)_{n\in\mathbb{N}} \subset \Omega$ such that $x_n \to x^*$. Then, by using the continuity of the kernel, we can show that $k(\cdot, x_n)$ is a Cauchy sequence:

$$\|k(\cdot, x_n) - k(\cdot, x_m)\|^2_{\mathcal{H}_k(\overline{\Omega})} = k(x_n, x_n) + k(x_m, x_m) - 2 \cdot k(x_n, x_m) \to 0.$$

As $\mathcal{H}_k(\Omega)$ is complete, there exists a unique limiting element $f \in \mathcal{H}_k(\Omega)$ such that $k(\cdot, x_n) \to f$ in $\mathcal{H}_k(\Omega)$, i.e. $f \in \overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}$. It is left to show that $f = k(\cdot, x^*)$, for which it suffices to show $f(x) = k(x, x^*)$ for all $x \in \Omega$:

$$\begin{aligned} |f(x) - k(x, x^*)| &= |f(x) - k(x, x_n) + k(x, x_n) - k(x, x^*)| \\ &\leq \|k(\cdot, x)\|_{\mathcal{H}_k(\Omega)} \cdot \|f - k(\cdot, x_n)\|_{\mathcal{H}_k(\Omega)} + |k(x, x_n) - k(x, x^*)|. \end{aligned}$$

The first summand vanishes due to $k(\cdot, x_n) \to f$ in $\mathcal{H}_k(\Omega)$, and the second summand vanishes by continuity of the kernel.
Therefore, we have $k(\cdot, x^*) = f$ and $f \in \overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}$, i.e. $k(\cdot, x^*) \in \overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}$.

□

Corollary 12 shows that whether $\mathcal{F}$ is closed or not, is just about $\Omega \subset \mathbb{R}^d$ being closed or not. In order to allow Ω to be also non-closed, we argue how to generalize the results from [23] to $\mathcal{F} \subset \Omega$ being precompact: By checking the proof of [23, Theorem 3.2] it becomes clear, that closedness of $\mathcal{F}$ is not required, only a decay on the Kolmogorov width $d_n(\mathcal{F})$. The closedness of $\mathcal{F}$ (and thus the compactness of $\mathcal{F}$) was likely only assumed, because this is a standard assumption in approximation theory. Indeed, [86, Theorem 1.1 (page 10)] shows that a precompact set and its closure provide the same Kolmogorov width. Therefore, it is convenient to always consider the corresponding closed and thus compact set.

Closedness of $\mathcal{F}$ might only be of interest if one strives for the maximal element which realizes the supremum of some continuous quantity – because the supremum needs not to be attained in open sets. But as the whole theory works also for weak algorithms, i.e. incorporating a $\gamma \in (0, 1)$, closedness of $\mathcal{F}$ is not required.

Similar to Corollary 12, also the reproducing kernel Hilbert spaces $\mathcal{H}_k(\Omega)$ and $\mathcal{H}_k(\overline{\Omega})$ related to Ω and $\overline{\Omega}$ are essentially the same:

Corollary 13. *Let $\Omega \subset \mathbb{R}^d$ be a bounded set and let $k : \overline{\Omega} \times \overline{\Omega} \to \mathbb{R}$ be a strictly positive definite continuous kernel.*
Then it follows that the embedding operator $E : \mathcal{H}_k(\Omega) \to \mathcal{H}_k(\overline{\Omega})$ (from Theorem 3) is surjective, i.e. $\mathcal{H}_k(\Omega)$ and $\mathcal{H}_k(\overline{\Omega})$ essentially contain the same functions.

Proof. In order to show that $E : \mathcal{H}_k(\Omega) \to \mathcal{H}_k(\overline{\Omega})$ is surjective, we consider $f \in \mathcal{H}_k(\overline{\Omega})$ and show that there is a function $\tilde{f} \in \mathcal{H}_k(\Omega)$ such that $E\tilde{f} = f$:
For this, we define $\tilde{f} := f|_\Omega$ which is an element of $\mathcal{H}_k(\Omega)$ by Theorem 4. For $x \in \Omega$ it holds $f(x) = \tilde{f}(x)$, therefore we consider $x \in \partial\Omega$ and a sequence $(x_n)_{n\in\mathbb{N}} \subset \Omega$ such that $x_n \to x$:

$$\begin{aligned} |f(x) - E\tilde{f}(x)| &\leq |f(x) - f(x_n) + f(x_n) - E\tilde{f}(x_n) + E\tilde{f}(x_n) - E\tilde{f}(x)| \\ &\leq |f(x) - f(x_n)| + |f(x_n) - E\tilde{f}(x_n)| + |E\tilde{f}(x_n) - E\tilde{f}(x)|. \end{aligned}$$

The first and third summand vanish for $n \to \infty$ because f respective $E\tilde{f}$ are elements of $\mathcal{H}_k(\overline{\Omega})$ and thus continuous. The second summand is equal to zero as $E\tilde{f}(x_n) = \tilde{f}(x_n) = f(x_n)$. As the space $\operatorname{span}\{k(\cdot, x), x \in \overline{\Omega}\}$ is dense in $\mathcal{H}_k(\overline{\Omega})$ (by construction), it follows $Ef|_\Omega = f$. □

3.3.2 Improved and generalized statements

This section is a mixture and extension of results within the abstract setting as introduced in [122] and [121]. We will frequently use the notion "any arbitrary selection rule which selects pairwise different elements": Like this, we want to allow for arbitrary sequences of pairwise different elements. The requirement on pairwise different elements is required because otherwise the Gram Schmidt orthogonalization procedure which was used in [23] and which will also be applied later on to prove the main abstract result (Theorem 14) does not work directly, but some extra technical work on circumventing this would be

required. In order to emphasize the use of the results for greedy algorithms, we stick to the notion of "arbitrary selection rule".

First, we start by generalizing Eq. (3.2) by introducing the quantity ν_n:

$$\begin{aligned} \sigma_n(\mathcal{F}) := \sigma_n(\mathcal{F})_{\mathcal{H}} &\equiv \sup_{f\in\mathcal{F}} \operatorname{dist}(f, V_n)_{\mathcal{H}}, \\ d_n(\mathcal{F}) := d_n(\mathcal{F})_{\mathcal{H}} &\equiv \inf_{Y_n\subset\mathcal{H}} \sup_{f\in\mathcal{F}} \operatorname{dist}(f, Y_n)_{\mathcal{H}}, \\ \nu_n := (\nu_n)_{\mathcal{H}} &\equiv \operatorname{dist}(f_n, V_n)_{\mathcal{H}}. \end{aligned} \tag{3.7}$$

This newly introduced quantity ν_n is not of much interest for the abstract analysis in [23] and was there simply denoted as $a_{n,n}$, but its consideration will pay off later on.

We have the following theorem, which is a unified formulation of [122, Theorem 1] and [121, Theorem 4] with the additional novelty that $\mathcal{F} \subset \Omega$ can be precompact. We remark that the precompactness of $\mathcal{F}$ is not required here, as $\mathcal{F}$ bounded is sufficient to keep $d_n(\mathcal{F})$ bounded. However, $\mathcal{F}$ precompact will be used later on, as it is equivalent to $d_n(\mathcal{F}) \to 0$ for $n \to \infty$, see [86, Prop. 1.2 (page 10)].

Theorem 14. *Consider a precompact set $\mathcal{F}$ in a separable Hilbert space $\mathcal{H}$ and a subset $\tilde{\mathcal{F}} \subseteq \mathcal{F}$ that contains infinitely many elements. Consider a greedy algorithm that selects pairwise different elements $\{f_0, f_1, ...\}$ from $\tilde{\mathcal{F}} \subseteq \mathcal{F}$ according to any arbitrary selection rule. We have the following inequalities between $\nu_n, \sigma_n(\tilde{\mathcal{F}})$ and $d_n(\mathcal{F})$ for any $N \geq 0, K \geq 1, 1 \leq m < K$:*

$$\prod_{i=1}^{K} \nu_{N+i}^2 \leq \left(\frac{K}{m}\right)^m \left(\frac{K}{K-m}\right)^{K-m} \sigma_{N+1}(\tilde{\mathcal{F}})^{2m} d_m(\mathcal{F})^{2K-2m}. \tag{3.8}$$

Both the proofs of [122, Theorem 1] and [121, Theorem 4] commented only on the minor changes necessary compared to the proof of the initial statement in [23, Theorem 3.2]. As we have here a more general statement, we include a full proof:

Proof. The proof is an extension of the proof of [23, Theorem 3.2], thus we proceed similar: As $\mathcal{H}$ is separable, we assume without loss of generality that $\mathcal{H}$ is $\ell^2(\mathbb{N} \cup \{0\})$. For the infinite sequence $(f_n)_{n\geq 0} \subseteq \tilde{\mathcal{F}} \subset \mathcal{H}$ consisting of pairwise different elements, we consider the orthonormal system $(f_n^*)_{n\geq 0}$ obtained by Gram-Schmidt orthogonalization. For the orthogonal projector $P_n : \mathcal{H} \to V_n \equiv \operatorname{span}\{f_0, ..., f_{n-1}\}$ it then holds

$$P_n f = \sum_{i=0}^{n-1} \langle f, f_i^* \rangle f_i^*.$$

Especially f_n can be expressed in this orthogonal basis, and we collect the coefficients in an (infinite dimensional) lower triangular matrix A:

$$A := (a_{i,j})_{i,j=0}^{\infty}, \; a_{i,j} := \langle f_i, f_j^* \rangle_{\mathcal{H}}.$$

Now we consider the $K \times K$ matrix $G = (g_{i,j})$ which is formed by the rows and columns of A with indices from $\{N+1, ..., N+K\}$. Each row $\mathbf{g}_i$ is the restriction of f_{N+i} to

the coordinates $N+1, ..., N+K$. Let $\mathcal{H}_m$ be the m-dimensional Kolmogorov subspace of $\mathcal{H}$ for which $\operatorname{dist}(\mathcal{F}, \mathcal{H}_m) = d_m$. Then, $\operatorname{dist}(f_{N+i}, \mathcal{H}_m) \leq d_m$, $i = 1, ..., K$. Let $\tilde{W}$ be the linear space which is the restriction of $\mathcal{H}_m$ to the coordinates $N+1, ..., N+K$. In general, $\dim(\tilde{W}) \leq m$. Let W be an m-dimensional space, $W \subset \operatorname{span}\{e_{N+1}, ..., e_{N+K}\}$, such that $\tilde{W} \subset W$ and P and $\tilde{P}$ are the orthogonal projections from $\mathbb{R}^K$ onto $W \subset \mathbb{R}^K$ and $\tilde{W} \subset \mathbb{R}^K$, respectively.

As the selection criterion chooses only elements from $\tilde{\mathcal{F}}$, it holds $f_i \in \tilde{\mathcal{F}}$ for all $i = 0, 1, ...$, and thus

$$\begin{aligned}\|P\mathbf{g}_i\|_{\mathbb{R}^K}^2 &\leq \|\mathbf{g}_i\|_{\mathbb{R}^K}^2 = \|f_{N+i} - \Pi_{V_{N+1}}(f_{N+i})\|_{\mathcal{H}}^2 \\ &\leq \sup_{f \in \tilde{\mathcal{F}}} \|f - \Pi_{V_{N+1}}(f)\|_{\mathcal{H}}^2 = \sigma_{N+1}(\tilde{\mathcal{F}})_{\mathcal{H}}^2 \quad i = 1, ..., K.\end{aligned}$$

Furthermore

$$\begin{aligned}\|\mathbf{g}_i - P\mathbf{g}_i\|_{\mathbb{R}^K} &\leq \|\mathbf{g}_i - \tilde{P}\mathbf{g}_i\|_{\mathbb{R}^K} = \operatorname{dist}(\mathbf{g}_i, \tilde{W}) \\ &\leq \operatorname{dist}(f_{N+i}, \mathcal{H}_m) \leq d_m(\mathcal{F}), \quad i = 1, ..., K.\end{aligned}$$

And finally

$$\begin{aligned}g_{i,i} \equiv a_{N+i,N+i} \quad &\equiv \|f_{N+i} - \Pi_{V_{N+i}} f_{N+i}\|_{\mathcal{H}} = \operatorname{dist}(f_{N+i}, V_{N+i})_{\mathcal{H}} \\ &\overset{\text{Eq. (3.7)}}{\equiv} \nu_{N+i}.\end{aligned}$$

Now Lemma 9 can be leveraged:

$$\begin{aligned}&\prod_{i=1}^{K} g_{i,i}^2 \leq \left\{\frac{1}{m}\sum_{i=1}^{K} \|P\mathbf{g}_i\|_2^2\right\}^m \left\{\frac{1}{K-m}\sum_{i=1}^{K} \|\mathbf{g}_i - P\mathbf{g}_i\|_2^2\right\}^{K-m} \\ \Rightarrow \quad &\prod_{i=1}^{K} \nu_{N+i}^2 \leq \left\{\frac{1}{m}\sum_{i=1}^{K} \sigma_{N+1}(\tilde{\mathcal{F}})_{\mathcal{H}}^2\right\}^m \left\{\frac{1}{K-m}\sum_{i=1}^{K} d_m(\mathcal{F})^2\right\}^{K-m} \\ \Leftrightarrow \quad &\prod_{i=1}^{K} \nu_{N+i}^2 \leq \left(\frac{K}{m}\right)^m \left(\frac{K}{K-m}\right)^{K-m} \sigma_{N+1}(\tilde{\mathcal{F}})^{2m} d_m(\mathcal{F})^{2K-2m}.\end{aligned}$$

□

We proceed by deriving two corollaries from Theorem 14: We start with Corollary 15, which is an extension of [121, Corollary 5] and will be leveraged in Section 3.4. Then we state Corollary 16, which is an extension of [122, Corollary 2] and will be leveraged in Section 3.5. In the following, log denotes the natural logarithm.

Corollary 15. *Under the assumptions of Theorem 14, but using the greedy selection criterion*

$$f_n \in \tilde{\mathcal{F}} \quad \textit{such that} \quad \operatorname{dist}(f_n, V_n)_{\mathcal{H}} \geq \gamma \cdot \sup_{f \in \tilde{\mathcal{F}}} \operatorname{dist}(f, V_n)_{\mathcal{H}} \tag{3.9}$$

for $\gamma \in (0, 1]$ it holds:

i) If $d_n(\mathcal{F})_{\mathcal{H}} \leq C_0 n^{-\alpha} \log(n)^{\delta}$ *with* $\alpha > 0, \delta \geq 0$ *for* $n = 1, 2, \ldots,$ *then it holds*

$$\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}} \leq C_1 n^{-\alpha} \log(n)^{\delta} \tag{3.10}$$

for $n = 8, 9, \ldots$ *with* $C_1 := 2^{5\alpha+1+\delta} C_0 \gamma^{-2}$.

ii) If $d_n(\mathcal{F})_{\mathcal{H}} \leq C_0 e^{-c_0 n^{\alpha}}$ *with* $\alpha > 0$ *for* $n = 1, 2, \ldots,$ *then it holds*

$$\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}} \leq \sqrt{2\tilde{C}_0} \gamma^{-1} e^{-c_1 n^{\alpha}} \tag{3.11}$$

for $n = 2, 3, \ldots$ *with* $\tilde{C}_0 := \max\{1, C_0\}$ *and* $c_1 = 2^{-(2+\alpha)} c_0 < c_0$.

We remark that for $\gamma = 1, \delta = 0$ this corollary simplifies to [121, Corollary 5].

Furthermore, we remark that statement ii) (and also the subsequent Corollary 16) could be generalized by adding a $\log(n)$ term to the exponent, i.e. assuming $d_n(\mathcal{F})_{\mathcal{H}} \leq C_0 e^{-c_0 \log(n)^{\beta} n^{\alpha}}$ for some $\beta \in \mathbb{R}$. However this does not provide much additional insight, therefore it is not included here. Instead, we point to [121, Corollary 11] for such a statement.

Proof. As $\nu_n \equiv \operatorname{dist}(f_n, V_n)_{\mathcal{H}}$, the selection criterion Eq. (3.9) means we have $\nu_n \geq \gamma \cdot \sup_{f \in \tilde{\mathcal{F}}} \operatorname{dist}(f, V_n)_{\mathcal{H}} \equiv \gamma \cdot \sigma_n(\tilde{\mathcal{F}})$. This can be plugged into Eq. (3.8) from Theorem 14:

$$\begin{aligned} &\prod_{i=1}^{K} \gamma^2 \sigma_{N+i}(\tilde{\mathcal{F}})^2 \leq \left(\frac{K}{m}\right)^m \left(\frac{K}{K-m}\right)^{K-m} \sigma_{N+1}(\tilde{\mathcal{F}})^{2m} d_m(\mathcal{F})^{2K-2m} \\ \Leftrightarrow \quad &\prod_{i=1}^{K} \sigma_{N+i}(\tilde{\mathcal{F}})^2 \leq \gamma^{-2K} \left(\frac{K}{m}\right)^m \left(\frac{K}{K-m}\right)^{K-m} \sigma_{N+1}(\tilde{\mathcal{F}})^{2m} d_m(\mathcal{F})^{2K-2m}. \end{aligned}$$

The remaining proof is very close to the proof of [23, Corollary 3.3]:

i) We consider $\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}}$ instead of $\sigma_n(\mathcal{F})_{\mathcal{H}}$, which is not a big change because the proof of [23, Corollary 3.3] is purely algebraic. However, we need to keep track of the logarithmic term $\log(n)^{\delta}$:

We use the monotonicity of $(\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}})_{n \geq 0}$ and use above's equation for $N = K = n$ and any $1 \leq m < n$ to obtain

$$\sigma_{2n}(\tilde{\mathcal{F}})^{2n} \leq \gamma^{-2n} \left(\frac{n}{m}\right)^m \left(\frac{n}{n-m}\right)^{n-m} \sigma_n(\tilde{\mathcal{F}})^{2m} d_m(\mathcal{F})^{2n-2m}.$$

We consider the case of n even, i.e. $n = 2s$ for some $s \geq 2$ and pick $m = s$. Thus the previous equation simplifies to

$$\begin{aligned} &\sigma_{4s}(\tilde{\mathcal{F}})^{4s} \leq \gamma^{-4s} 4^s \sigma_{2s}(\tilde{\mathcal{F}})^{2s} d_s(\mathcal{F})^{2s} \\ \Leftrightarrow \quad &\sigma_{4s}(\tilde{\mathcal{F}}) \leq \sqrt{2} \gamma^{-1} \sqrt{\sigma_{2s}(\tilde{\mathcal{F}}) d_s(\mathcal{F})}. \end{aligned} \tag{3.12}$$

We proceed by assuming the contradiction and consider M the first value such that $\sigma_M(\tilde{\mathcal{F}}) > C_1 M^{-\alpha} \log(M)^{\delta}$.

- First case: $M = 4s$. From Eq. (3.12) using $\sigma_{2s}(\tilde{\mathcal{F}}) \leq C_1(2s)^{-\alpha}\log(2s)^{\delta}$ and the decay assumption on $d_s(\mathcal{F})$, we have

$$\begin{aligned}\sigma_{4s}(\tilde{\mathcal{F}}) &\leq \sqrt{2}\gamma^{-1}\sqrt{(C_1(2s)^{-\alpha}\log(2s)^{\delta})(C_0 s^{-\alpha}\log(s)^{\delta})} \\ &= \sqrt{2}\sqrt{C_1C_0 2^{\delta-\alpha}}\gamma^{-1}\log(s)^{\delta}s^{-\alpha},\end{aligned}$$

  whereby we used $\log(2s) \leq 2\log(s)$ for $s \geq 2$. It follows

$$C_1(4s)^{-\alpha}\log(4s)^{\delta} < \sigma_{4s}(\tilde{\mathcal{F}}) \leq \sqrt{2}\sqrt{C_1C_0 2^{\delta-\alpha}}\gamma^{-1}\log(s)^{\delta}s^{-\alpha},$$

  which can be rearranged to

$$\begin{aligned} C_1 &\leq 4^{\alpha}\sqrt{2}\sqrt{C_1C_0 2^{\delta-\alpha}}\gamma^{-1}\frac{\log(s)^{\delta}}{\log(4s)^{\delta}} \\ &< 2^{2\alpha+(\delta-\alpha+1)/2}\sqrt{C_1C_0}\gamma^{-1} \\ \Leftrightarrow \quad C_1 &< 2^{3\alpha+1+\delta}C_0\gamma^{-2} < 2^{5\alpha+1+\delta}C_0\gamma^{-2},\end{aligned}$$

  which is the desired contradiction to the definition of C_1.

- Second case: $M = 4s + q, q \in \{1, 2, 3\}$. Using Eq. (3.12) and the monotonicity of $(\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}})_{n\geq 0}$ gives

$$\begin{aligned} C_1 2^{-\alpha}(4s)^{-\alpha}\log(s)^{\delta} &< C_1(4s+q)^{-\alpha}\log(4s+q)^{\delta} \\ &< \sigma_{4s+q}(\tilde{\mathcal{F}}) \leq \sigma_{4s}(\tilde{\mathcal{F}}) \\ &\leq \sqrt{2}\sqrt{C_1C_0 2^{\delta-\alpha}}\gamma^{-1}\log(s)^{\delta}s^{-\alpha} \\ \Rightarrow C_1 &< 2 < 2^{5\alpha+\delta+1}C_0\gamma^{-2},\end{aligned}$$

  which is again the desired contradiction.

ii) The proof of [23, Corollary 3.3] is slightly inaccurate, because its Eq. (3.9) uses

$$\sigma_{2n+1} \leq \sqrt{2C_0}\gamma^{-1}e^{-c_0 2^{-1-\alpha}(2n)^{\alpha}},$$

however

$$\sigma_{2n+1} \leq \sqrt{2C_0}\gamma^{-1}e^{-c_0 2^{-1-\alpha}(2n+1)^{\alpha}}$$

would be required.
Thus, instead we use ii) of Corollary 16 (which is possible because the proof of Corollary 16 does not rely on Corollary 15) for our selection criterion from Eq. (3.9), i.e. it holds $\nu_n \equiv \operatorname{dist}(f_n, V_n)_{\mathcal{H}} \geq \gamma \cdot \sigma_n(\tilde{\mathcal{F}})$.
Using the monotonicity of $(\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}})_{n\geq 0}$, Eq. (3.14) gives the result:

$$\gamma \cdot \sigma_n(\tilde{\mathcal{F}}) \leq \left(\prod_{i=n+1}^{2n} \gamma \cdot \sigma_i(\tilde{\mathcal{F}})\right)^{1/n} \leq \left(\prod_{i=n+1}^{2n} \nu_i\right)^{1/n} \leq \sqrt{2\tilde{C}_0} \cdot e^{-c_1 n^{\alpha}}.$$

□

Corollary 16. *Under the assumptions of Theorem 14 (i.e. using any arbitrary selection rule that selects pairwise different elements):*

i) If $d_n(\mathcal{F}) \leq C_0 n^{-\alpha} \log(n)^\delta$ *with* $\alpha > 0, \delta \geq 0$ *for* $n = 1, 2, ...,$ *then it holds*

$$\left(\prod_{i=n+1}^{2n} \nu_i \right)^{1/n} \leq 2^{\alpha+\delta+1/2} \tilde{C}_0 e^\alpha \cdot \log(n)^{\alpha+\delta} n^{-\alpha}, \tag{3.13}$$

for $n = 3, 4, ...$ *with* $\tilde{C}_0 := \max\{1, C_0\}$.

ii) If $d_n(\mathcal{F}) \leq C_0 e^{-c_0 n^\alpha}$ *with* $\alpha > 0$ *for* $n = 1, 2, ...,$ *then it holds*

$$\left(\prod_{i=n+1}^{2n} \nu_i \right)^{1/n} \leq \sqrt{2\tilde{C}_0} \cdot e^{-c_1 n^\alpha}, \tag{3.14}$$

for $n = 2, 3, ...$ *with* $\tilde{C}_0 := \max\{1, C_0\}$ *and* $c_1 = 2^{-(2+\alpha)} c_0 < c_0$.

Proof. The proof is a minor adjustment of the proof of [122, Corollary 2], because we track the additional δ dependent terms:
We exactly follow its proof until including its Eq. (12), which reads

$$\left(\prod_{i=1}^{n} \nu_{n+i} \right)^{1/n} \leq \sqrt{2} \cdot d_m(\mathcal{F})^{(n-m)/n} \qquad \text{for any } 1 \leq m < n,$$

To prove statement i) and ii), we proceed slightly different:

i) We follow the lines of the proof of [122, Corollary 2] and additionally keep track of the parameters δ and γ: For n fixed we choose a fixed $0 < \omega \ll 1$ and define $m^* := \lceil \omega n \rceil \in \mathbb{N}$, i.e. $\omega n \leq m^* < \omega n + 1$. Using $d_n(\mathcal{F}) \leq 1$, $d_n(\mathcal{F}) \leq \tilde{C}_0 n^{-\alpha} \log(n)^\delta$ with $\tilde{C}_0 := \max\{1, C_0\}$, and since $d_n(\mathcal{F})$ is non-increasing, we can estimate:

$$\begin{aligned}
\left(\prod_{i=1}^{n} \nu_{n+i} \right)^{1/n} &\leq \sqrt{2} \cdot d_{m^*}(\mathcal{F})^{(n-m^*)/n} \\
&\leq \sqrt{2} \cdot d_{\lceil \omega n \rceil}(\mathcal{F})^{(n-\omega n-1)/n} \\
&\leq \sqrt{2} \tilde{C}_0^{(1-\omega)-1/n} \lceil \omega n \rceil^{-\alpha(1-\omega)+\alpha/n} \lceil \log(\omega n) \rceil^{\delta(1-\omega)-\delta/n} \\
&\leq \sqrt{2} \tilde{C}_0 (\omega n)^{-\alpha(1-\omega)} (\omega n)^{\alpha/n} \lceil \log(\omega n) \rceil^{\delta(1-\omega)}.
\end{aligned}$$

The last inequality made use of $-\alpha(1-\omega) + \alpha/n = -\alpha(1-\omega-1/n) < 0$, such that $\lceil \omega n \rceil \geq \omega n$ could be used. Using $(\omega n)^{\alpha/n} \leq (n^{1/n})^\alpha \leq 2^\alpha$ we reformulate this as

$$\begin{aligned}
\left(\prod_{i=1}^{n} \nu_{n+i} \right)^{1/n} &\leq 2^{\alpha+1/2} \tilde{C}_0 \omega^{-\alpha(1-\omega)} n^{-\alpha(1-\omega)} \lceil \log(\omega n) \rceil^{\delta(1-\omega)} \\
&\leq 2^{\alpha+1/2} \tilde{C}_0 \omega^{-\alpha} n^{\alpha\omega} \lceil \log(\omega n) \rceil^{\delta(1-\omega)} n^{-\alpha}.
\end{aligned}$$

Now we select $\omega = 1/\log(n) \in (0,1)$ for $n \geq 3$. With this choice, we obtain for the ω-dependent factors:

$$\begin{aligned}\omega^{-\alpha} n^{\alpha\omega} \lceil \log(\omega n) \rceil^{\delta(1-\omega)} &= \log(n)^{\alpha} (n^{1/\log(n)})^{\alpha} \lceil \log(n/\log(n)) \rceil^{\delta\left(1-\frac{1}{\log(n)}\right)} \\ &= \log(n)^{\alpha} e^{\alpha} \lceil \log(n) - \log(\log(n)) \rceil^{\delta\left(1-\frac{1}{\log(n)}\right)} \\ &\leq \log(n)^{\alpha} e^{\alpha} \lceil \log(n) \rceil^{\delta} \leq \log(n)^{\alpha} e^{\alpha} (2\log(n))^{\delta} \\ &= e^{\alpha} 2^{\delta} \log(n)^{\alpha+\delta}.\end{aligned}$$

Thus overall we obtain

$$\left(\prod_{i=1}^{n} \nu_{n+i} \right)^{1/n} \leq 2^{\alpha+1/2} \tilde{C}_0 e^{\alpha} 2^{\delta} \log(n)^{\alpha+\delta} n^{-\alpha},$$

which is the desired bound.

ii) Here we can directly reuse the proof from [122, Corollary 2], as even the statement is exactly the same.

□

Corollary 15 and Corollary 16 look quite similar, and also their proofs are similar. Thus it would be desirable to unify the statements. However, when using any arbitrary selection rule as in Corollary 16, the convergence is slowed down by an additional $\log(n)^{\alpha}$ term, which does not pop up when using a (weak) P-greedy selection criterion as in Corollary 15. Indeed, the additional $\log(n)^{\alpha}$ term can be removed by bounding the error with help of entropy numbers instead of Kolmogorov widths. This was done in [94] during the finalization of this thesis.

3.4 Analysis of kernel approximation using greedy algorithms

In this section we use the results of the previous Section 3.3, especially Corollary 15 to obtain results for general kernel interpolation. In particular, we leverage greedy algorithms as a tool to prove general kernel interpolation statements. The main statement of this subsection is, that convergence rates (in the number of interpolation points) for kernel interpolation in $\mathcal{H}_k(\Omega)$ do not deteriorate if we consider the RKHS over a smaller subset, i.e. $\mathcal{H}_{\tilde{k}}(\tilde{\Omega})$ with $\tilde{\Omega} \subseteq \Omega \subset \mathbb{R}^d$. This is formalized in Theorem 17. These results were mostly covered in the publication [121].

We start by considering two sets $\tilde{\Omega} \subseteq \Omega \subset \mathbb{R}^d$, where $\tilde{\Omega}$ is usually the set of interest and Ω is a well-behaved domain such that we have approximation statements for $\mathcal{H}_k(\Omega)$. Furthermore, we choose a continuous strictly positive definite kernel $k : \Omega \times \Omega \to \mathbb{R}$ with $k(x,x) \leq 1$ for all $x \in \Omega$. This restriction is not too severe, because it can be enforced by normalizing the kernel, i.e.

$$k'(x,y) := \frac{k(x,y)}{\sqrt{k(x,x)} \cdot \sqrt{k(y,y)}},$$

as long as $\sup_{x\in\Omega} k(x,x) < \infty$. In order to leverage the framework from the previous subsection, we define

$$\begin{aligned}\mathcal{H} &:= \mathcal{H}_k(\Omega),\\ \tilde{\mathcal{F}} &:= \{k(\cdot,x), x\in\tilde{\Omega}\} \subset \mathcal{H},\\ \mathcal{F} &:= \{k(\cdot,x), x\in\Omega\} \subset \mathcal{H},\end{aligned} \tag{3.15}$$

and recall that it holds $V_n = V(X_n)$, see Eq. (3.3). We make use of the detailed notation from Section 2.2, especially of the restricted kernel $\tilde{k}$ from Eq. (2.6) and also include the used kernel and the considered domain in the power function in the following, i.e. e.g. P_{k,Ω,X_n}. Furthermore for the projectors, e.g. $\Pi_{\mathcal{H}_k(\Omega),V(X_n)}$ we here also include the RKHS, where the projection takes place, in the notation, i.e. $\Pi_{\mathcal{H}_k(\Omega),V(X_n)} : \mathcal{H}_k(\Omega) \to V(X_n)$. Using these choices, we can compute the quantities of Corollary 15:

$$\begin{aligned}\sigma_n(\tilde{\mathcal{F}})_{\mathcal{H}} &= \sup_{f\in\tilde{\mathcal{F}}} \operatorname{dist}(f,V_n)_{\mathcal{H}} = \sup_{f\in\tilde{\mathcal{F}}} \|f - \Pi_{V_n}(f)\|_{\mathcal{H}}\\ &= \sup_{x\in\tilde{\Omega}} \|k(\cdot,x) - \Pi_{\mathcal{H}_k(\Omega),V(X_n)}(k(\cdot,x))\|_{\mathcal{H}_k(\Omega)}\\ &= \sup_{x\in\tilde{\Omega}} P_{k,\Omega,X_n}(x) = \sup_{x\in\tilde{\Omega}} P_{\tilde{k},\tilde{\Omega},X_n}(x) = \|P_{\tilde{k},\tilde{\Omega},X_n}\|_{L^\infty(\tilde{\Omega})}, \qquad (3.16)\\ d_n(\mathcal{F})_{\mathcal{H}} &\equiv \inf_{Y_n\subset\mathcal{H}} \sup_{f\in\mathcal{F}} \operatorname{dist}(f,Y_n)_{\mathcal{H}} = \inf_{Y_n\subset\mathcal{H}} \sup_{f\in\mathcal{F}} \|f - \Pi_{Y_n}(f)\|_{\mathcal{H}}\\ &\leq \inf_{Y_n\subset\mathcal{F}} \sup_{f\in\mathcal{F}} \|f - \Pi_{Y_n}(f)\|_{\mathcal{H}_k(\Omega)} = \inf_{X_n\subset\Omega} \|P_{k,\Omega,X_n}\|_{L^\infty(\Omega)} \qquad (3.17)\end{aligned}$$

With these choices, the selection criterion from Eq. (3.9) can be stated as

$$x_{n+1} \in \tilde{\Omega} \text{ such that } \quad P_{\tilde{k},\tilde{\Omega},X_n}(x_{n+1}) \geq \gamma \cdot \|P_{\tilde{k},\tilde{\Omega},X_n}\|_{L^\infty(\Omega)}$$

for $X_n := \{x_1,...,x_n\}$ and $\gamma \in (0,1]$. This is also called the *weak P-greedy algorithm*, see also Definition 18.

Now we can state the main result, which is a small extension of [121, Theorem 7] by adding a possible $\log(n)^\delta$ decay rate:

Theorem 17. *Let $\tilde{\Omega} \subset \mathbb{R}^d$ be arbitrary. If there exists a bounded superset $\Omega \supseteq \tilde{\Omega}$ and a sequence of (non-necessarily nested) sets of points $(X_n)_{n\in\mathbb{N}} \subset \Omega$ such that for some $\alpha > 0, \delta \geq 0$ it holds*

1. *(algebraic decay case)*

$$\|f - \Pi_{\mathcal{H}_k(\Omega),V(X_n)} f\|_{L^\infty(\Omega)} \leq C_0 n^{-\alpha} \log(n)^\delta \cdot \|f\|_{\mathcal{H}_k(\Omega)} \quad \forall f \in \mathcal{H}_k(\Omega), \tag{3.18}$$

then the weak P-greedy algorithm with $\gamma \in (0,1]$ using $\tilde{k}$ applied to $\tilde{\Omega}$ gives a nested sequence of sets of points $(\tilde{X}_n)_{n\in\mathbb{N}} \subset \tilde{\Omega}$ such that

$$\|f - \Pi_{\mathcal{H}_{\tilde{k}}(\tilde{\Omega}),\tilde{V}(\tilde{X}_n)} f\|_{L^\infty(\tilde{\Omega})} \leq C_1 n^{-\alpha} \log(n)^\delta \cdot \|f\|_{\mathcal{H}_{\tilde{k}}(\tilde{\Omega})} \quad \forall f \in \mathcal{H}_{\tilde{k}}(\tilde{\Omega})$$

for $n = 8, 9, ...$ with $C_1 = 2^{5\alpha+1+\delta} C_0 \gamma^{-2}$.

2. (exponential decay case)

$$\|f - \Pi_{\mathcal{H}_k(\Omega),V(X_n)} f\|_{L^\infty(\Omega)} \leq C_0 e^{-c_0 n^\alpha} \cdot \|f\|_{\mathcal{H}_k(\Omega)} \quad \forall f \in \mathcal{H}_k(\Omega), \tag{3.19}$$

then the weak P-greedy algorithm with $\gamma \in (0,1)$ using $\tilde{k}$ applied to $\tilde{\Omega}$ gives a nested sequence of sets of points $(\tilde{X}_n)_{n\in\mathbb{N}} \subset \tilde{\Omega}$ such that

$$\|f - \Pi_{\mathcal{H}_{\tilde{k}}(\tilde{\Omega}),\tilde{V}(\tilde{X}_n)} f\|_{L^\infty(\tilde{\Omega})} \leq \sqrt{2\tilde{C}_0}\gamma^{-1} e^{-c_1 n^\alpha} \cdot \|f\|_{\mathcal{H}_{\tilde{k}}(\tilde{\Omega})} \quad \forall f \in \mathcal{H}_{\tilde{k}}(\tilde{\Omega})$$

for $n = 2, 3., ...$ with $\tilde{C}_0 := \max\{1, C_0\}$ and $c_1 = 2^{-(2+\alpha)} c_0 < c_0$.

Proof. We want to leverage Corollary 15 and therefore need to check its requirements: As k is continuous and $\Omega \subset \mathbb{R}^d$, the RKHS $\mathcal{H}_k(\Omega)$ is separable. The selection criterion of the weak P-greedy algorithm with $\gamma \in (0,1]$ using $\tilde{k}$ applied to $\tilde{\Omega}$ equals exactly the selection criterion from Corollary 15, i.e. Eq. (3.9). Using Eq. (3.17), the prerequisites Eq. (3.18) and Eq. (3.19) give bounds on the Kolmogorov widths $d_n(\mathcal{F})_\mathcal{H}$ as

$$d_n(\mathcal{F})_\mathcal{H} \leq \inf_{X_n \subset \Omega} \|P_{k,\Omega,X_n}\|_{L^\infty(\Omega)} \leq \begin{cases} C_0 n^{-\alpha} \log(n)^\delta & \text{algebraic decay case} \\ C_0 e^{-c_0 n^\alpha} & \text{exponential decay case.} \end{cases}$$

The set $\mathcal{F}$ is precompact, because $\overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}$ is compact which follows from being closed, bounded, $d_n(\overline{\mathcal{F}}^{\|\cdot\|_{\mathcal{H}_k(\Omega)}}) = d_n(\mathcal{F}) \overset{n\to\infty}{\longrightarrow} 0$ and the application of [86, Prop. 1.2 (page 10)]. Thus the application of Corollary 15 yields

$$\|P_{\tilde{k},\tilde{\Omega},X_n}\|_{L^\infty(\tilde{\Omega})} \overset{\text{Eq. (3.16)}}{=} \sigma_n(\tilde{\mathcal{F}})_\mathcal{H} \leq \begin{cases} C_1 n^{-\alpha} \log(n)^\delta & \text{algebraic decay case} \\ \sqrt{2\tilde{C}_0}\gamma^{-1} e^{-c_1 n^\alpha} & \text{exponential decay case,} \end{cases}$$

with the constants as specified in the theorem. Using the supremum formulation of the power function from Eq. (2.11) immediately yields the result. □

Theorem 17 can be used to derive kernel approximation statements for non-standard domains: The idea is to start with a well shaped domain $\Omega \subset \mathbb{R}^d$, for which convergence results are available. Then, leveraging Theorem 17, these convergence rates also transfer to any subdomain $\tilde{\Omega} \subset \Omega$. This can be used to derive upper bounds on sampling numbers, to show a monotonicity of convergence rates with respect to domains and to derive convergence rates on non-Lipschitz domains. Details are elaborated in [121, Section 4.3] and numerical results are provided in [121, Section 5].

As an extension compared to [121], we also included a possible $\log(n)^\delta$ rate for the algebraic case. This is of interest for e.g. tensor product kernels, which usually give rise to mixed smoothness spaces. Such tensor product kernels are non-radial, i.e. the provided theory is general enough to hold beyond radial kernels.

3.5 Analysis of greedy algorithms in kernel setting

In this section we use the results of the previous Section 3.3, especially Corollary 16 to obtain convergence rates for a scale of greedy algorithms, including the greedy kernel

algorithms introduced in the literature, i.e. the P-greedy, f-greedy, f/P-greedy and $f \cdot P$-greedy. This main result is formalized in Corollary 22. These results were mostly covered in the publication [122].

We start by introducing the unifying scale of β-greedy algorithms in Section 3.5.1 and proceed with an analysis of those in Section 3.5.2.

3.5.1 Unifying scale of greedy kernel algorithms: β-greedy

We generalize the definition of β-greedy algorithms from [122, Definition 4] to weak β-greedy algorithms. The notion *weak* refers to the fact, that one does not pick the maximizing element as it was done e.g. in Eq. (2.25), instead it is sufficient to pick any element that is above a certain threshold.

Definition 18. *A greedy kernel algorithm on $\Omega \subset \mathbb{R}^d$ is called weak β-greedy algorithm with parameters $\beta \in [0, \infty], \gamma \in (0, 1]$, if the next interpolation point $x_{n+1} \in \Omega$ is chosen to satisfy:*

$$\eta^{(n)}(x_{n+1}) \geq \gamma \cdot \sup_{x \in \Omega} \eta^{(n)}(x)$$

with

$$\eta^{(n)}(x) = \begin{cases} |(f - s_n)(x)|^\beta \cdot P_n(x)^{1-\beta} & \beta \in [0, 1] \\ |(f - s_n)(x)| \cdot P_n(x)^{1/\beta - 1} & \beta \in (1, \infty) \\ \frac{|(f-s_n)(x)|}{P_n(x)} & \beta = \infty. \end{cases}$$

First, we remark that for $\beta \in (1, \infty)$, we used a slightly different scaling of the selection criterion compared to [122]. However, as elaborated in [122, Remark 5], this different scaling does not affect the selection condition. However, here it was used in order to obtain a nice and consistent behavior in the additional γ parameter, which was not used in [122]. Second, we recall that these selection criteria do not yield a deterministic sequence of points $(x_n)_{n \in \mathbb{N}}$, especially for $\gamma < 1$ usually several points satisfy the selection criterion. However, the forthcoming analysis holds for any such sequence of points.
The scale of greedy algorithms is visualized in Figure 3.1, which was taken from [122, Figure 1]: It can be seen that the special choices of $\beta = 0$, $\beta = 1/2$, $\beta = 1$ and the limiting case $\beta \to \infty$ give respectively the P-greedy, the $f \cdot P$-greedy, f-greedy and f/P-greedy algorithm.

As elaborated in [122], these β-greedy algorithms are well-defined for $\beta \in [0, \infty)$. For $\beta = \infty$, i.e. the f/P-greedy, one should formally consider a weak selection with $\gamma < 1$ due to the counter example in [119, Example 6].

3.5.2 Unifying analysis of β-greedy kernel algorithms

To start with the analysis, we recall [122, Lemma 6], where we added that we consider pairwise distinct points. We remark that the statement holds for any strictly positive definite kernel k (that can be normalized to satisfy $k(x, x) \leq 1$ for all $x \in \Omega$ as discussed in Section 3.4) on any set Ω, because it is based solely on RKHS theory.

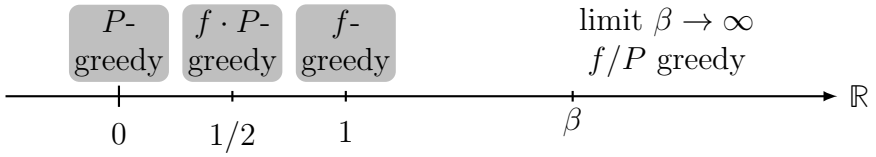

Figure 3.1: Visualization of the scale of the β-greedy algorithms on the real line. Several important cases for $\beta \in \{0, 1/2, 1\}$ and $\beta \to \infty$ are marked. Figure taken from [122].

Lemma 19. *For any sequence $\{x_i\}_{i\in\mathbb{N}} \subset \Omega$ of pairwise distinct points and any $f \in \mathcal{H}_k(\Omega)$ it holds for all $n = 1, 2, \ldots$ that*

$$\left[\prod_{i=n+1}^{2n} |r_i(x_{i+1})|\right]^{1/n} \leq n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})\right]^{1/n}, \tag{3.20}$$

whereby $r_{i+1} \equiv f - s_{i+1}$ is the residual.

In order to connect the quantity $|r_i(x_{i+1})|$ to $\|r_i\|_{L^\infty(\Omega)}$, we recall [122, Lemma 7]. Recall that a different scaling in β was used in Definition 18 for $\beta \in (1, \infty)$, therefore the statement slightly differs from [122, Lemma 7].

Lemma 20. *Any weak β-greedy algorithm with $\beta \in [0, \infty]$ and admissible*[1] *$\gamma \in (0, 1]$ applied to a function $f \in \mathcal{H}_k(\Omega)$ satisfies for $i = 0, 1, \ldots$:*

a) In the case of $\beta \in [0, 1]$:

$$\|r_i\|_{L^\infty(\Omega)} \leq \gamma^{-1} \cdot |r_i(x_{i+1})|^\beta \cdot P_i(x_{i+1})^{1-\beta} \cdot \|r_i\|^{1-\beta}_{\mathcal{H}_k(\Omega)}. \tag{3.21}$$

b) In the case of $\beta \in (1, \infty]$ with $1/\infty := 0$:

$$\|r_i\|_{L^\infty(\Omega)} \leq \gamma^{-1} \cdot \frac{|r_i(x_{i+1})|}{P_i(x_{i+1})^{1-1/\beta}} \cdot \|P_i\|^{1-1/\beta}_{L^\infty(\Omega)}. \tag{3.22}$$

Proof. We reuse the proof of [122, Lemma 7] and incorporate the use of *weak* greedy algorithms, i.e. the occuring parameter $\gamma \in (0, 1]$. Furthermore, the different scaling in β for $\beta \in (1, \infty)$ compared to [122] gives slightly modified formulae:

a) For $\beta = 0$, i.e. the weak P-greedy algorithm, Eq. (3.21) is the standard power function estimate Eq. (2.13) in conjunction with the P-greedy selection criterion $P_n(x_{n+1}) \geq \gamma \cdot \|P_n\|_{L^\infty(\Omega)}$:

$$\begin{aligned} \|r_i\|_{L^\infty(\Omega)} &\leq \|P_{X_i}\|_{L^\infty(\Omega)} \cdot \|r_i\|_{\mathcal{H}_k(\Omega)} \\ &\leq \gamma^{-1} \cdot P_{X_i}(x_{i+1}) \cdot \|r_i\|_{\mathcal{H}_k(\Omega)}. \end{aligned}$$

For $\beta = 1$, Eq. (3.21) is simply the selection criterion of the weak f-greedy since we have here $r_n(x_{n+1}) \geq \gamma \cdot \|r_n\|_{L^\infty(\Omega)}$.

[1] The notion "admissible" refers to the restriction, that the choice $\gamma = 1$ is not allowed for $\beta = \infty$ due to the counter example in [119, Example 6].

We thus consider $\beta \in (0,1)$ and let $\tilde{x}_{i+1} \in \Omega$ be such that $|r_i(\tilde{x}_{i+1})| = \|r_i\|_{L^\infty(\Omega)}$. Then the selection criterion from Definition 18 gives

$$\gamma \cdot |r_i(x)|^\beta \cdot P_i(x)^{1-\beta} \leq |r_i(x_{i+1})|^\beta \cdot P_i(x_{i+1})^{1-\beta} \quad \forall x \in \Omega,$$

and in particular for $x = \tilde{x}_{i+1}$

$$P_i(\tilde{x}_{i+1}) \leq \gamma^{-\frac{1}{1-\beta}} \frac{|r_i(x_{i+1})|^{\frac{\beta}{1-\beta}}}{|r_i(\tilde{x}_{i+1})|^{\frac{\beta}{1-\beta}}} \cdot P_i(x_{i+1}).$$

Using this bound with the standard power function estimate of Eq. (2.13) gives

$$\begin{aligned}
\|r_i\|_{L^\infty(\Omega)} = |r_i(\tilde{x}_{i+1})| &\leq P_i(\tilde{x}_{i+1}) \cdot \|r_i\|_{\mathcal{H}_k(\Omega)} \\
&\leq \gamma^{-\frac{1}{1-\beta}} \cdot \frac{|r_i(x_{i+1})|^{\frac{\beta}{1-\beta}}}{|r_i(\tilde{x}_{i+1})|^{\frac{\beta}{1-\beta}}} \cdot P_i(x_{i+1}) \cdot \|r_i\|_{\mathcal{H}_k(\Omega)} \\
&= \gamma^{-\frac{1}{1-\beta}} \cdot \frac{|r_i(x_{i+1})|^{\frac{\beta}{1-\beta}}}{\|r_i\|_{L^\infty(\Omega)}^{\frac{\beta}{1-\beta}}} \cdot P_i(x_{i+1}) \cdot \|r_i\|_{\mathcal{H}_k(\Omega)} \\
\Leftrightarrow \|r_i\|_{L^\infty(\Omega)}^{\frac{1}{1-\beta}} &\leq \gamma^{-\frac{1}{1-\beta}} \cdot |r_i(x_{i+1})|^{\frac{\beta}{1-\beta}} \cdot P_i(x_{i+1}) \cdot \|r_i\|_{\mathcal{H}_k(\Omega)}.
\end{aligned}$$

Taking the exponential $(\cdot)^{1-\beta}$ yields the final result.

b) For $\beta \in (1, \infty)$, the selection criterion Definition 18 can be rearranged to

$$|r_i(x)| \leq \gamma^{-1} \cdot \frac{|r_i(x_{i+1})|}{P_i(x_{i+1})^{1-1/\beta}} \cdot P_i(x)^{1-1/\beta} \quad \forall x \in \Omega \setminus X_i,$$

and taking the supremum $\sup_{x \in \Omega \setminus X_i}$ gives (after bounding the corresponding power function value with $\|P_i\|_{L^\infty(\Omega)}$):

$$\|r_i\|_{L^\infty(\Omega)} \leq \gamma^{-1} \cdot \frac{|r_i(x_{i+1})|}{P_i(x_{i+1})^{1-1/\beta}} \cdot \|P_i\|_{L^\infty(\Omega)}^{1-1/\beta}.$$

For $\beta = \infty$, the same steps can be done to directly yield the statement (when using the notation $1/\infty = 0$).

□

Now, Lemma 20 can be used to estimate the $|r_i(x_{i+1})|$ quantities appearing in Equation (3.20). This then yields the next Theorem 21, which is a slight generalization of [122, Theorem 8]:

Theorem 21. *Any weak β-greedy algorithm with $\beta \in [0, \infty]$ and admissible $\gamma \in (0, 1]$ applied to a function $f \in \mathcal{H}_k(\Omega)$ satisfies the following error bound for $n = 1, 2, \ldots$:*

a) In the case of $\beta \in [0, 1]$:

$$\left[\prod_{i=n+1}^{2n} \|r_i\|_{L^\infty(\Omega)}\right]^{1/n} \leq \gamma^{-1} \cdot n^{-\beta/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})\right]^{1/n}. \tag{3.23}$$

b) In the case of $\beta \in (1, \infty]$ with $1/\infty := 0$:

$$\left[\prod_{i=n+1}^{2n} \|r_i\|_{L^\infty(\Omega)}\right]^{1/n} \leq \gamma^{-1} \cdot n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})^{1/\beta}\right]^{1/n}. \tag{3.24}$$

Proof. We reuse the proof of [122, Theorem 8] and incorporate the use of *weak* greedy algorithms and the slightly modified scaling in β. We prove the two cases separately:

a) For $\beta = 0$, i.e. the weak P-greedy, Eq. (3.21) gives $\|r_i\|_{L^\infty(\Omega)} \leq \gamma^{-1} \cdot P_i(x_{i+1}) \cdot \|r_i\|_{\mathcal{H}_k(\Omega)}$. Taking the product $\prod_{i=n+1}^{2n}$ and the n-th root in conjunction with the estimate $\|r_i\|_{\mathcal{H}_k(\Omega)} \leq \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}$ for $i = n+1, \ldots, 2n$ gives the result.

For $\beta \in (0, 1]$, we start by reorganizing the estimate from Eq. (3.21) of Lemma 20 to get

$$|r_i(x_{i+1})| \geq \gamma^{1/\beta} \cdot \left(\|r_i\|_{L^\infty(\Omega)}^{1/\beta}\right) / \left(P_i(x_{i+1})^{\frac{1-\beta}{\beta}} \cdot \|r_i\|_{\mathcal{H}_k(\Omega)}^{\frac{1-\beta}{\beta}}\right),$$

and we use this to bound the left-hand side of Eq. (3.20) as

$$\begin{aligned}
n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})\right]^{1/n} &\geq \left[\prod_{i=n+1}^{2n} |r_i(x_{i+1})|\right]^{1/n} \\
&\geq \gamma^{1/\beta} \cdot \left[\prod_{i=n+1}^{2n} \left(\|r_i\|_{L^\infty(\Omega)}^{1/\beta}\right) / \left(P_i(x_{i+1})^{\frac{1-\beta}{\beta}} \cdot \|r_i\|_{\mathcal{H}_k(\Omega)}^{\frac{1-\beta}{\beta}}\right)\right]^{1/n} \\
&= \gamma^{1/\beta} \cdot \left[\prod_{i=n+1}^{2n} \|r_i\|_{L^\infty(\Omega)}^{1/\beta}\right]^{1/n} \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})^{\frac{1-\beta}{\beta}} \cdot \|r_i\|_{\mathcal{H}_k(\Omega)}^{\frac{1-\beta}{\beta}}\right]^{-1/n}.
\end{aligned}$$

Rearranging the factors, and using again the fact that $\|r_i\|_{\mathcal{H}_k(\Omega)} \leq \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}$ for $i = n+1, \ldots, 2n$, gives

$$\begin{aligned}
&\gamma^{1/\beta} \cdot \left[\prod_{i=n+1}^{2n} \|r_i\|_{L^\infty(\Omega)}^{1/\beta}\right]^{1/n} \\
&\leq n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})^{1/\beta}\right]^{1/n} \cdot \left[\prod_{i=n+1}^{2n} \|r_i\|_{\mathcal{H}_k(\Omega)}^{\frac{1-\beta}{\beta}}\right]^{1/n} \\
&\leq n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})^{1/\beta}\right]^{1/n} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}^{\frac{1-\beta}{\beta}} \\
&\leq n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}^{1/\beta} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1})^{1/\beta}\right]^{1/n}.
\end{aligned}$$

Now, the inequality can be raised to the exponent β to give the final statement.

b) For $\beta \in (1, \infty]$ we proceed similarly by first rewriting Eq. (3.22) of Lemma 20 as

$$|r_i(x_{i+1})| \geq \gamma \cdot \left(\|r_i\|_{L^\infty(\Omega)} \cdot P_i(x_{i+1})^{1-1/\beta}\right) / \left(\|P_i\|_{L^\infty(\Omega)}^{1-1/\beta}\right),$$

and we lower bound the left-hand side of Eq. (3.20) as

$$n^{-1/2}\cdot\|r_{n+1}\|_{\mathcal{H}_k(\Omega)}\cdot\left[\prod_{i=n+1}^{2n}P_i(x_{i+1})\right]^{1/n}\geq\left[\prod_{i=n+1}^{2n}|r_i(x_{i+1})|\right]^{1/n}$$
$$\geq\gamma\cdot\left[\prod_{i=n+1}^{2n}\left(\|r_i\|_{L^\infty(\Omega)}\cdot P_i(x_{i+1})^{1-1/\beta}\right)/\left(\|P_i\|_{L^\infty(\Omega)}^{1-1/\beta}\right)\right]^{1/n}.$$

Rearranging for $\left[\prod_{i=n+1}^{2n}\|r_i\|_{L^\infty(\Omega)}\right]^{1/n}$ yields

$$\left[\prod_{i=n+1}^{2n}\|r_i\|_{L^\infty(\Omega)}\right]^{1/n}$$
$$\leq\gamma^{-1}n^{-1/2}\cdot\|r_{n+1}\|_{\mathcal{H}_k(\Omega)}\cdot\left[\prod_{i=n+1}^{2n}\|P_i\|_{L^\infty(\Omega)}^{1-1/\beta}\right]^{1/n}\cdot\left[\prod_{i=n+1}^{2n}P_i(x_{i+1})^{1/\beta}\right]^{1/n},$$

which gives the final result due to $\|P_i\|_{L^\infty(\Omega)}\leq 1$ for all $i=0,1,..$.

□

In order to obtain useful bounds from Theorem 21, it is required to bound the geometric mean of subsequent power function values, i.e. $[\prod_{i=1}^{2n}P_i(x_{i+1})]^{1/n}$. This can be done with help of the abstract setting and in particular Corollary 16. Similar to Section 3.4 we choose again

$$\mathcal{H}:=\mathcal{H}_k(\Omega),$$
$$\mathcal{F}:=\{k(\cdot,x),x\in\Omega\}\subset\mathcal{H},$$

and recall that it holds $V_n=V(X_n)$, see Eq. (3.3). We do not make use of $\tilde{\Omega}$ here anymore and only use Ω, in order to keep the argumentation clear. However, in order to derive target data-dependent convergence rates also for non-Lipschitz domains, the analysis of Section 3.4 and Section 3.5 can be easily combined.

As done in Eq. (3.16) and Eq. (3.17), we compute or estimate the quantities of Corollary 15:

$$\begin{aligned}\sigma_n(\mathcal{F})_{\mathcal{H}}&=\|P_{X_n}\|_{L^\infty(\Omega)},\\ d_n(\mathcal{F})_{\mathcal{H}}&\leq\inf_{X_n\subset\Omega}\|P_{X_n}\|_{L^\infty(\Omega)}.\end{aligned}\tag{3.25}$$

Furthermore, for later on, we need the following statement on the quantity ν_n of Corollary 16:

$$\begin{aligned}\nu_n&\equiv\operatorname{dist}(f_n,V_n)_{\mathcal{H}}=\|f_n-\Pi_{V_n}(f)\|_{\mathcal{H}}\\ &=\|k(\cdot,x_{n+1})-\Pi_{V(X_n)}(k(\cdot,x_{n+1})\|_{\mathcal{H}_k(\Omega)}=P_{X_n}(x_{n+1}).\end{aligned}\tag{3.26}$$

With these connections at hand, we can derive the following result, which is a slight generalization of [122, Corollary 11]:

Corollary 22. *Assume that a weak β-greedy algorithm with $\beta \in [0,\infty]$ and admissible $\gamma \in (0,1]$ is applied to a function $f \in \mathcal{H}_k(\Omega)$. Let $\alpha > 0$, $\delta \geq 0$ and $C_0, c_0 > 0$ be given constants, and set $1/\infty := 0$. Recall $r_i \equiv f - s_i$:*

1. *If there exists a sequence $(X_n)_{n\in\mathbb{N}} \subset \Omega$ of sets of points such that*

$$\left\| \tilde{f} - \Pi_{V(X_n)} \tilde{f} \right\|_{L^\infty(\Omega)} \leq C_0 n^{-\alpha} \log(n)^\delta \cdot \|\tilde{f}\|_{\mathcal{H}_k(\Omega)} \quad \forall \tilde{f} \in \mathcal{H}_k(\Omega),$$

then for all $\beta \geq 0$ and for all $n \geq 8$ it holds

$$\begin{aligned} &\min_{n+1\leq i\leq 2n} \|r_i\|_{L^\infty(\Omega)} \\ \leq & C\gamma^{-1} \cdot n^{-\frac{\min\{1,\beta\}}{2} - \frac{\alpha}{\max\{1,\beta\}}} \cdot \log(n)^{\frac{\delta+\alpha}{\max\{1,\beta\}}} \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}, \end{aligned} \tag{3.27}$$

with $C := \left(2^{\alpha+1/2} \max\{1, C_0\} e^\alpha\right)^{\frac{1}{\max\{1,\beta\}}}$.

2. *If there exists a sequence $(X_n)_{n\in\mathbb{N}} \subset \Omega$ of sets of points such that*

$$\left\| \tilde{f} - \Pi_{V(X_n)} \tilde{f} \right\|_{L^\infty(\Omega)} \leq C_0 e^{-c_0 n^\alpha} \|\tilde{f}\|_{\mathcal{H}_k(\Omega)} \quad \forall \tilde{f} \in \mathcal{H}_k(\Omega),$$

then for all $\beta \geq 0$ and for all $n \geq 2$ it holds

$$\min_{n+1\leq i\leq 2n} \|r_i\|_{L^\infty(\Omega)} \leq C\gamma^{-1} \cdot n^{-\frac{\min\{1,\beta\}}{2}} e^{-c_1 n^\alpha} \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}, \tag{3.28}$$

with $C := \left(\sqrt{2\max\{1, C_0\}}\right)^{\frac{1}{\max\{1,\beta\}}}$ and $c_1 = 2^{-(2+\alpha)} c_0 / \max\{1,\beta\}$.

3. *For f/P-greedy, for any kernel and for all $n \geq 1$ it holds*

$$\min_{n+1\leq i\leq 2n} \|r_i\|_{L^\infty(\Omega)} \leq \gamma^{-1} n^{-1/2} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)}.$$

Proof. We mostly reuse the proof of [122, Corollary 11] and incorporate the use of *weak* greedy algorithms, i.e. the occuring parameter $\gamma \in (0,1]$ as well as the adjusted scaling in β for $\beta \in (1,\infty)$: The proof is a simple combination of Corollary 15 and Theorem 21, with the addition of the following simple steps:

First, the provided worst case bounds for functions in $\mathcal{H}_k(\Omega)$ (either algebraic or exponential) imply the same bound on the power function via Eq. (2.10), (2.11). Second, in all cases we use the results of Theorem 21 in combination with the bound

$$\min_{i=n+1,\dots,2n} \|r_i\|_{L^\infty(\Omega)} \leq \left[\prod_{i=n+1}^{2n} \|r_i\|_{L^\infty(\Omega)} \right]^{1/n}.$$

Then, Eq. (3.23) and (3.24) of Theorem 21 can be jointly written as

$$\left[\prod_{i=n+1}^{2n} \|r_i\|_{L^\infty(\Omega)} \right]^{1/n} \leq n^{-\frac{\min\{1,\beta\}}{2}} \cdot \|r_{n+1}\|_{\mathcal{H}_k(\Omega)} \cdot \left[\prod_{i=n+1}^{2n} P_i(x_{i+1}) \right]^{\frac{1}{n\max\{1,\beta\}}}.$$

The last factor can be estimated with help of Corollary 16 by using the connection $\nu_i = P_i(x_{i+1})$ established in Eq. (3.26). This gives the result of the first two points.

The third point directly follows from Eq. (3.24) for $\beta = \infty$ due to $P_i(x_{i+1}) \leq 1$ for all $i = 1, 2, \dots$. □

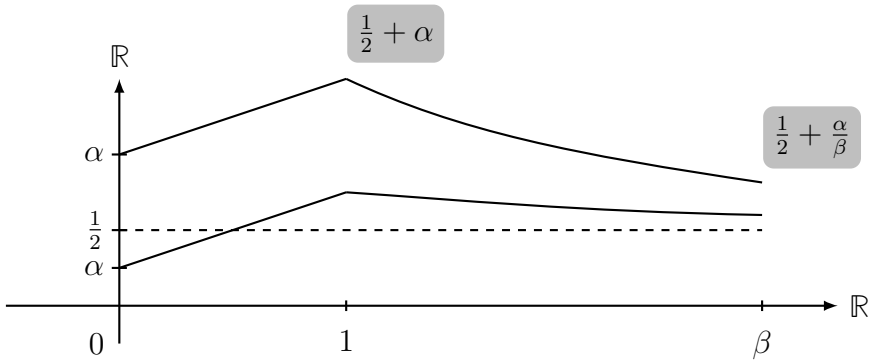

Figure 3.2: Visualization of the algebraic main decay rate $n^{-\frac{\min\{1,\beta\}}{2}-\frac{\alpha}{\max\{1,\beta\}}}$ (y-axis) from Corollary 22 for $\alpha \in \{0.25, 1\}$ in dependence on the β parameter (x-axis).

We remark that for the P-greedy algorithm (i.e. $\beta = 0$) the error bounds hold for $\|r_{2n}\|_{L^\infty(\Omega)}$, i.e. without using the minimum $\min_{n+1\leq i\leq 2n}$. It remains unclear whether this minimum within the formulation of the results is really necessary, or whether a monotonically decaying error bound can be proven.

The assumptions of Corollary 22 are satisfied in various cases: If the kernel is an RBF kernel of finite smoothness with some decay in the Fourier transform, then the RKHS is norm equivalent to a Sobolev space (see Corollary 5) and thus sampling inequalities can be leveraged (see Theorem 7). More details on this can be found in [122, Section 5.2].

3.5.3 Numerical example

Although the results in this section are a significant step in the analysis of target data-dependent greedy algorithm, the derived convergence rates still seem to be suboptimal yet: For this we recall the example from [122, Section 6.2], which made use of the Wendland "$k = 0$" kernel

$$k(x, y) = \max(1 - |x - y|, 0)$$

on the domain $\Omega = [0, 1]$. It holds $\mathcal{H}_k(\Omega) \asymp H^1(\Omega)$ such that we have $\alpha = 1/2$ and $\delta = 0$ within Corollary 22. Due to the piecewise linear nature of the kernel k, kernel interpolation boils down to piecewise linear interpolation, which allows also leveraging classical results from spline interpolation theory. The function $f : \Omega \to \mathbb{R}, x \to x^\omega$ for $1/2 < \omega < 1$ is considered. The assumption $\omega > 1/2$ ensures that $f \in H^1(\Omega) \asymp \mathcal{H}_k(\Omega)$. Furthermore, we recall that for asymptotically uniformly distributed interpolation points – i.e. $q_{X_n} \asymp h_{X_n,\Omega} \asymp n^{-1}$ – it holds

$$\|f - s_n\|_{L^\infty(\Omega)} \geq C_\omega \cdot n^{-\omega}, \tag{3.29}$$

while for any distribution of points (i.e. also optimally distributed points) it holds

$$\|f - s_n\|_{L^\infty(\Omega)} \geq C \cdot n^{-2}. \tag{3.30}$$

Figure 3.3 visualizes the numerical decay behavior of the $\|\cdot\|_{L^\infty(\Omega)}$ error for various values of $\beta \in [0, \infty]$. In view of the convergence rates provided by Corollary 22, we have the following comments:

- P-greedy: Here, the error decays only slowly and together with Eq. (3.29) the proven decay rate due to Corollary 22 seems to be optimal.

- f-greedy: Here we observe the fastest convergence among all the displayed methods. Corollary 22 gives a main decay rate of n^{-1}, however numerically one can observe a faster convergence rate. Indeed, the lower bound Eq. (3.30) allows for way faster convergence, and Figure 3.3 indicates that f-greedy seems to realize this optimal decay rate in this example.

- β-greedy for intermediate β values: For intermediate values of β, i.e. $\beta \in (0, 1)$ and $\beta \in (1, \infty)$ we observe numerically intermediate convergence rates. However similar to what was remarked for f-greedy, the rates provided by Corollary 22 do not seem to be sharp yet.

Thus, this example shows that the provided data-dependent convergence rates do not yet seem to be sharp, which leaves space for future research.

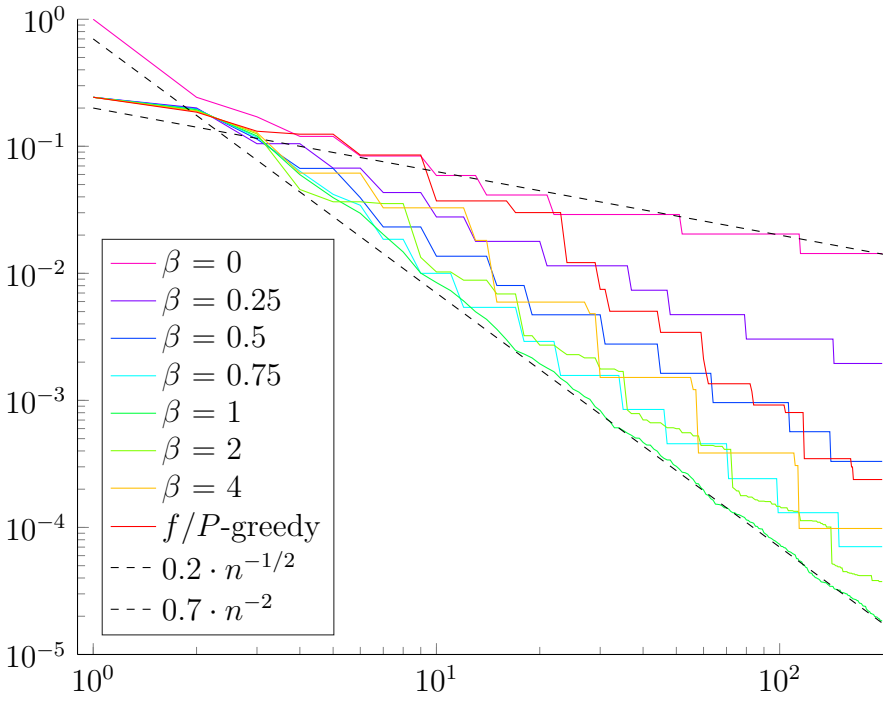

Figure 3.3: Decay of the error $\|f - s_n^{(\beta)}\|_{L^\infty(\Omega)}$ (y-axis) for β-greedy algorithms in the number n of chosen interpolation points (x-axis) for $\beta \in \{0, 0.25, 0.5, 0.75, 1, 2, 4, \infty\}$ and $f(x) = x^\omega$ with $\omega = 0.51$. Two additional dashed lines indicate a rate of convergence of $n^{-1/2}$ and n^{-2}. Figure taken from [122] and slightly adapted to the β-parametrization in this thesis.

3.6 Analysis of generalized greedy kernel methods

The framework presented in Section 3.2 to Section 3.5 is not restricted to kernel interpolation. Indeed, it is possible to generalize the results to generalized kernel interpolation, see Section 2.4: First, the β-greedy algorithms are extended to generalized β-greedy algorithms. Then the convenient connection from Eq. (3.15) can be reestablished. Furthermore the Lemmata and Theorems from Section 3.5 can be generalized. Finally, worst case optimal bounds can be used to derive the same rates of convergence for the greedy algorithms, with faster convergence rates for target data-dependent greedy algorithms. We stress that this is indeed a generalization of the theory: If one chooses the generalized interpolation functionals to be point evaluation functionals, then the theory boils down again to the theory presented in Section 3.5. The details were presented in the publication [123].

This extended theory can be used to approximate the solution of PDEs: Consider a suitable linear elliptic PDE of a suitable domain, such as the Laplace equation

$$\begin{aligned} -\Delta u &= f, \\ u|_{\partial\Omega} &= g \end{aligned}$$

with suitable functions f, g. Then one can consider point evaluation functionals δ_x for $x \in \partial\Omega$ as well as point evaluations of the Laplacian, i.e. $\delta_x \circ (-\Delta)$ for $x \in \Omega$. The important and appealing point here is, that the target data-dependency of the greedy algorithms refers to the functions f, g and *not* to the unknown solution u. Thus, an adaptive choice of collocation points can be computed based on the right-hand side functions of the PDE, which are usually known.

3.7 Discussion, comments and outlook

In Section 3.1 it was explained that for the analysis of greedy kernel algorithms no strong results were available till the analysis provided in [92], which established the connections Eq. (3.4) and Eq. (3.5) between the abstract analysis of greedy algorithms in Hilbert spaces and the analysis of greedy kernel algorithms. Indeed, this connection proved to be quite fruitful: By modifying, generalizing and extending the abstract analysis (see Section 3.2), it was possible to reuse these connections to derive the line of results presented in Section 3.3, Section 3.4, Section 3.5 and Section 3.6. One key step in the derivation of target data-dependent convergence rates was to bound the quantity

$$\left(\prod_{i=n+1}^{2n} \nu_i \right)^{1/n} = \left(\prod_{i=n+1}^{2n} P_i(x_{i+1}) \right)^{1/n}, \tag{3.31}$$

and to focus on minimum errors $\min_{i=1,\dots,2n} \|f - s_i\|_{L^\infty(\Omega)}$ instead of $\|\cdot\|_{L^\infty(\Omega)}$.

Preliminary results (not presented here) indeed indicate that not only the geometric mean of subsequent power function values from Eq. (3.31) decays independent of the sequence $(x_i)_{i\in\mathbb{N}} \subset \Omega$, but also

$$\left(\prod_{i=n+1}^{2n} |\langle f, v_i \rangle_{\mathcal{H}_k(\Omega)}| \right)^{1/n} \tag{3.32}$$

decays independent of the choice of the sequence $(x_i)_{i\in\mathbb{N}} \subset \Omega$, whereby $\{v_i\}$ is the Newton basis (see e.g. Eq. (2.26)). As it holds

$$\|f - s_n\|_{L^\infty(\Omega)} = P_n(x_{n+1}) \cdot |\langle f, v_n\rangle_{\mathcal{H}_k(\Omega)}|$$

for the f-greedy selection criterion, the observation that the quantity from Eq. (3.32) decays, might allow for a refined analysis. However, it was not possible to derive an upper bound on Eq. (3.32).

The analysis presented in this Chapter 3 focussed on the $\|\cdot\|_{L^\infty(\Omega)}$ norm. This makes sense, as the power function is naturally linked to the $\|\cdot\|_{L^\infty(\Omega)}$ norm and also the reproducing property gives point evaluations, which are closer to the $\|\cdot\|_{L^\infty(\Omega)}$ norm than to any other norm. Nevertheless, it would be of interest to generalize the analysis to other norms, such as the $\|\cdot\|_{L^2(\Omega)}$ norm. The trivial estimate

$$\|f - s_n\|_{L^2(\Omega)} \leq |\Omega| \cdot \|f - s_n\|_{L^\infty(\Omega)}$$

allows transfering any $\|\cdot\|_{L^\infty(\Omega)}$ rate to a $\|\cdot\|_{L^2(\Omega)}$ rate. Indeed, simple numerical experiments (not shown here) indicate that for non-uniformly distributed f-greedy points one cannot in general expect faster $\|\cdot\|_{L^2(\Omega)}$ convergence rates. On the other hand, for well distributed points, the $L^2(\Omega)$ error frequently decays faster than the $L^\infty(\Omega)$ error, see e.g. Duchon's localization principle [30, 111]. A recent analysis of $\|\cdot\|_{L^2(\Omega)}$ convergence rates was done in [115], which showed that the $L^2(\Omega)$ convergence rates using asymptotically uniformly distributed points are sharp for the *escaping the native space* case, i.e. for functions $f \in \mathcal{C}(\Omega)$ of smoothness less or equal to $\mathcal{H}_k(\Omega)$. Improved upper bounds on $L^2(\Omega)$ convergence rates using asymptotically uniformly distributed points were derived in [91] for the (continuous) superconvergence case. However, the question on optimality of the derived convergence rates for adaptively chosen points remains open: Generally speaking, it remains for future research whether it is possible to optimize for a specific norm or even for a specific quantity of the residual $f - s_n$, i.e. devising "goal oriented selection criteria" [26].

The line of analysis established through [23, 92, 121, 122] leveraged so far upper bounds on sampling numbers. These upper bounds on sampling numbers are usually obtained by using an equivalence to Sobolev spaces and/or directly by using sampling inequalities [39, 76, 77, 113] or (local) fill distance based arguments [129], see [112, Section 11] for an overview. Here it would be of interest, how to generalize this to immediately obtain bounds on sampling numbers from the kernel or its Mercer expansion directly – without relying on Fourier transform arguments. In view of the Fourier decay of translational invariant kernels from Eq. (2.2), where upper bounds usually give error estimates while lower bounds give stability results, it seems likely that similar statements should also hold with respect to upper and lower bounds on the eigenvalue decay within the Mercer expansion. Promising results and tools in this direction were recently established in [59,60] or [25]. Another approach could be possible by using [7], where the spectral asymptotics of the determinant of the kernel matrix in the flat limit is analyzed. By using the expression of the power function as ratio of determinants via Eq. (2.12), the monotonicity of the power function as $P_{X_N}(x) \leq P_{X_n}(x)$ and using

$$k(\|\epsilon x - \epsilon y\|) = \Phi(\|\epsilon x - \epsilon y\|) = \Phi(\epsilon\|x - y\|) = k_\epsilon(x, y)$$

the results of [7] can be used to bound the worst case error for point sets $\epsilon X_n = \{\epsilon x_1, ..., \epsilon x_n\}$.

Further analysis also needs to be devoted to analytic kernels, such as the Gaussian kernel or e.g. inverse multiquadric kernels: For these kernels, the standard tools due to the frequent norm equivalence of the RKHS to Sobolev spaces cannot be reused. Future research topics in this direction include a better understanding of escaping the native space and superconvergence, but also whether and when an adaptive point choice is beneficial: In case of exponential convergence rates, the additional convergence rate of $n^{-1/2}$ within Corollary 22 only seems to be beneficial for high-dimensional problems, as it does not suffer from the curse of dimensionality.

Chapter 4

Deep kernel methods

In contrast to Chapter 3, this section puts more emphasis on the development of new methods and algorithms rather than analyzing already existing ones. As discussed in Chapter 1, around ten years ago the field of deep learning and especially deep neural networks has resurged. Nowadays, these methods tend to outperform standard kernel methods in several tasks, especially tasks dealing with very high dimensional and huge data sets. This success is mainly based on leveraging the unprecedented availability of data in conjunction with the possibility of using increasing computational power. While learning with large amounts of data is an issue for kernel methods, representation learning is not available at all: Using a fixed kernel "off the shelves" means using a fixed feature map. However, kernel methods still provide advantages in specific tasks where they outperform neural networks, and they are appealing due to their rich analytic background. Thus, it is quite natural to ask about the interplay of these two types of machine learning methods and how to make use of each other: Is it possible to combine both approaches to derive enhanced models? A promising idea to incorporate the advantages of neural networks into kernel methods is to make use of the multilayer setup of neural networks. Furthermore, it is possible to leverage common neural network strategies also for kernel methods, e.g. for the optimization of kernels. A theoretical foundation for the analysis of multilayer kernel methods was provided in [13], which proved a deep kernel representer theorem, see Theorem 8. Using this deep kernel representer theorem and applying ideas from machine learning, we will derive two different kind of deep kernel models.

In Section 4.1, related literature is reviewed and the the state of the art is discussed. In Section 4.2, Structured Deep Kernel Networks (SDKNs) are introduced, which are built on top of the aforementioned deep kernel representer theorem. Their construction, benefits and approximation properties are discussed. Subsequently in Section 4.3, another deep kernel model is introduced and analyzed, namely *two-layered kernels*. Especially their use in conjunction with greedy algorithms is highlighted, resulting in the 2L-VKOGA algorithm. Finally Section 4.4 discusses and comments these results while also giving an outlook.

4.1 Literature and state of the art

In the last decades, several approaches have been developed to combine the benefits of kernels with the benefits of neural networks or more general multilayer methods, especially to obtain some kind of feature learning. This subsection first discusses methods from the literature, which are relevant for the approach of structured deep kernel networks (SDKNs) presented in Section 4.2. These methods mainly revolve around introducing new models which make use of kernels. Subsequently, relevant methods for the two-layered kernel approach of Section 4.3 are discussed: Here, the approach sticks more closely to the use of a base kernel, which is adapted to the data.

In order to obtain improved features of the data (representation learning), models combining neural networks with kernels have been proposed. All those approaches have in common, that some sort of neural network is used for the representation learning, and that the kernel model is used for the final prediction (regression/classification). We discuss the following approaches more or less in chronological order:

The references [67, 68] introduce convolutional kernel networks in both a supervised and unsupervised way. Doing so, they derive a kernel acting on images with a kernel feature map based on the convolutional mappings of a convolutional neural network. In [61], a neural network is trained such that the in-class similarity of pairs of data is maximized, while the similarity for instances of different classes is minimized. The approach proposed in [125] makes use of a neural network which transforms the data, and of a Gaussian process on top for the final prediction. Both the front end (neural network) and kernel model (back end) with all their hyperparameters are optimized jointly. The study [3] uses a related approach, i.e. using a kernel model on top of a neural network, however they do not optimize the neural network at all anymore: Focussing on image data, they simply extract features from a pretrained convolutional neural network and train a kernel model on top of these features leveraging a Nyström approximation. With this approach, they achieve comparable accuracy compared to also fine-tuning the neural network, but save several orders of magnitude of training time.

An approach that tries to handle deep forward neural networks and deep kernel machines in a joint framework was introduced via a conjugate feature duality in [106], whereby the conjugated features can be interpreted as hidden features. By coupling existing kernel machines, deep kernel models (so-called *deep restricted kernel machines* (deep RKM)) are build. This framework of deep RKMs is general enough to also allow for e.g. generative models (Gen-RKMs) [80, 81], where (convolutional) neural networks are used to define kernels in a primal way.

There are also models that do not use neural networks as feature extractor, but imitate their architecture using kernels: [130] introduces *Deep Spectral Kernel Learning*, which is a concatenation of kernel approximation mappings, namely random fourier features. This can be seen as a neural network with fixed parameters in every second layer, where the weights are fixed and given by random fourier features. Furthermore, there are approaches that enhance neural networks by optimizing their activation functions: From the theoretical point of view, [109] derives a representer theorem for activation functions, which are then given by nonuniform linear splines. A connection to splines and sparsity is also provided, which is related to the discussion of the optimality of the SDKNs introduced in

Section 4.2. The representer theorem in [109] was enabled by considering an extended loss functional with a specific regularization term that is motivated by favorable properties of activation functions which work in practice. The employed proof is related to the proof of the deep kernel representer theorem in [13]. Another approach was introduced in [96], namely *Kafnets*. Here, they assume every activation function to be a (one dimensional) kernel model based on a few preselected centers within each layer, e.g. a grid. This approach is related to the SDKNs introduced later on in Section 4.2, however there, the centers will be given by the deep kernel representer theorem and thus yield optimality of the approach.

Without relying on neural networks too much, the following approaches still put kernels into a multilayer setup: In the Gaussian process community, deep Gaussian processes were introduced in [20]. Using approximation techniques, they were scaled to large datasets.

Recently much work was also devoted to the *neural tangent kernel* (NTK) [53] (see also Section 2.6.1) because it was found that for specific scaling limits (where the ratio between width and depth of the network stays lower bounded away from zero) the NTK shows a non-trivial evolution [50, 131]. Thus, this could be interpreted as a kernel with data-dependent learned features, though it does not allow for a closed form expression. However, such approaches are not treated in the following.

In Section 4.3, a two-layered kernel approach is introduced. In contrast to the approach within Section 4.2, this approach can be interpreted as a modification of a base kernel. A simple modification of a base kernel, in particular of RBF kernels, can be done due to the shape parameter $\varepsilon > 0$ from Eq. (2.1). Several works focussed on selecting or optimizing such a single hyperparameter for RBFs, see e.g. [112] or [34, §14] for an overview or [29, 35, 71] for some detailed works regarding the shape parameter. Such a single shape parameter, that scales the distance behavior within the considered data is suitable, as long as there is no specific direction-dependent behavior within the data. In order to address such cases, anisotropic kernels were considered (see e.g. [34, Section 3]) that use one shape parameter per Euclidean direction of the data. This approach poses two challenges: First, it is no longer amenable to cross-validation for already medium dimensional datasets due to the curse of dimensionality. Second, it is only suitable if the Euclidean directions are of special significance – as no other directions can be identified. Therefore, there have been developed algorithms for this general challenge of *distance metric learning*, see e.g. [40, 41, 103] in general or [2, 14, 21, 88] in the RBF context. Other approaches for predicting suitable shape parameters or detecting relevant features make use of neural networks respective neural network like techniques [74, 78]. Another approach, which is of particular interest for Section 4.3, was introduced under the notion *kernel flows* [79] and further used in [48]. This approach can be seen as a non-parametric family of very deep kernels: Given a base kernel, this kernel is modified by incremental changes applied to its inputs. This sequence of incremental changes deforms the kernel and gives raise to the notion of a very deep kernel, which is thus able to learn features from the data. The optimization method which was used for designing these kernels used e.g. cross-validation criteria based on a maximal Lyapunov exponent or Maximum Mean Discrepancy (MMD).

Another approach of relevance for the following is given by *active subspaces* [17, 18]: These are subspaces along which the target function varies. This is in contrast to inactive subspaces, where the target function does not vary, and which are therefore not of interest.

For the determination of these active subspaces, originally gradient information was used, which is however usually not available in applications.

4.2 Structured Deep Kernel Networks

This subsection is devoted to the introduction of Structured Deep Kernel Networks (SDKNs), which use the deep kernel representer theorem from Theorem 8 in a principled way to put kernels in a multilayer setup. We show that the obtained SDKNs are powerful models with strong analytic properties, which nevertheless fit into the framework of the representer theorem. This mostly covers the results, which were presented in [120].

We start by motivating the need for a *structured* setup of deep kernel networks in Section 4.2.1, then continue with the construction of the SDKN with help of simple classes of base kernels in Section 4.2.2. Section 4.2.3 briefly discusses the optimization of this model, and Section 4.2.4 then elaborates on the approximation, universality and optimality properties of the SDKNS. Section 4.2.5 presents the relation to neural networks, while Section 4.2.6 concludes with a discussion.

4.2.1 Failure of straightforward deep RBF networks

The deep kernel representer theorem from Theorem 8 could be used in conjunction with matrix-valued radial basis function kernels, which are frequently used in applications, i.e. kernels like

$$k(x, y) = k_{\text{Gauss}} \cdot I_d \in \mathbb{R}^{d \times d}$$

for suitable values of the dimension $d \in \mathbb{N}$. However, numerical experiments (not presented here) did not show a good performance of such an ansatz, even for toy data sets. The main obstacle is likely the abundance of parameters, as for a dataset of N centers, the ansatz provided by Theorem 8 consists of $\sum_{l=1}^{L} d_l \cdot N = N \cdot \sum_{l=1} d_l \gg N$ parameters. This is way more than the number of data points, hence overfitting problems are very likely.

In order to alleviate this problem, the sparsification motivation of the greedy algorithms from 3.5 can be employed, i.e. one considers only $M \ll N$ centers which yields overall $\sum_{l=1}^{L} d_l \cdot M$ parameters. Using such an approach, every mapping $f_l : \mathbb{R}^{d_l} \to \mathbb{R}^{d_{l+1}}$ is given by

$$f_l(\cdot) = \sum_{i=1}^{M} k_l(\cdot, F_{l-1}(z_i))\alpha_i \tag{4.1}$$

with optimizable parameters $\alpha_i \in \mathbb{R}^{d_{l+1}}$. Nevertheless, also this approach did not yield convincing performance in numerical experiments, and the optimization was at least one of the bottlenecks here: Assume the output of the function f_l should be slightly adjusted, s.t. for $F_{l-1}(z_i)$ it holds

$$\begin{aligned} & f_l(F_{l-1}(z_j)) = y_j + \epsilon \\ \text{instead of} \quad & f_l(F_{l-1}(z_j)) = y_j, \end{aligned}$$

with $\epsilon \in \mathbb{R}^{d_{l+1}}$. The coefficients α_i, $i = 1, ..., M$ from Eq. (4.1) are linked to the outputs y_j respective $y_j + \epsilon$ via the kernel matrix $A \in \mathbb{R}^{Md_l \times Md_l}$, $A_{ij} = k_l(F_{l-1}(z_i), F_{l-1}(z_j))$ – which however may be severely ill-conditioned or even singular depending on the mapping F_{l-1}. Such an ill-conditioning can yield large changes in the parameters α_j, $j = 1, ..., M$ even for small values of ϵ – which can be very challenging for gradient descent optimization procedures [66]. A standard way to circumvent this would be to change from the basis of kernel translates to a Newton or Lagrange basis. However, these bases depend on the centers which are given by $F_{l-1}(z_i)$ and thus dependent on the previous mappings – i.e. they had to be recomputed again and again which is infeasible.

To summarize, the main obstacle of this straightforward approach arises from the radiality of the kernels, which do not allow for a disentangling of different features. Hence, the next subsection introduces structured deep kernel networks, which make use of different kind of kernels in a structured setup.

4.2.2 Construction of SDKNs

The construction of SDKNs relies on two classes of kernels. The first one is the matrix-valued linear kernel k_{lin}, which is defined as

$$\begin{aligned} k_{\text{lin}} : \quad \mathbb{R}^d \times \mathbb{R}^d &\to \mathbb{R}^{b \times b}, \\ (x, y) &\mapsto \langle x, y \rangle_{\mathbb{R}^d} \cdot I_b. \end{aligned} \tag{4.2}$$

The linear kernel will be used to allow linear combinations of features. We recall the following statement from [120, Proposition 2]:

Proposition 23. *The mapping $\mathbb{R}^d \to \mathbb{R}^b, x \mapsto Ax$ with $A \in \mathbb{R}^{b \times d}$ can be realized as a kernel mapping*

$$s : \mathbb{R}^d \to \mathbb{R}^b, x \mapsto \sum_{i=1}^{M} k_{lin}(x, z_i)\alpha_i, \quad \alpha_i \in \mathbb{R}^b$$

with given centers $\{z_i\}_{i=1}^M \subset \mathbb{R}^d$ by using the matrix-valued linear kernel from Eq. (4.2), iff the span of the center points is a superset of the row space of the matrix A.

The proof is given in Section A.1. We proceed with the following statement [120, Proposition 3], which shows that the requirement "iff the span of the center points is a superset of the row space" is not that restrictive

Proposition 24. *Consider a mapping $g : \mathbb{R}^{d_0} \supset \Omega \to \{(g^{(1)}(x), .., g^{(d)}(x))^\top \mid x \in \Omega\} =: g(\Omega) \subset \mathbb{R}^d, x \mapsto (g^{(1)}(x), .., g^{(d)}(x))^\top$ with $g^{(1)}, .., g^{(d)} : \Omega \to \mathbb{R}$.*
Then any linear combination

$$(g^{(1)}(x), .., g^{(d)}(x))^\top \mapsto \sum_{j=1}^{d} g^{(j)}(x) \cdot \beta_j$$

with $\beta_j \in \mathbb{R}^b, j = 1, .., d$ can be realized with help of a linear kernel k_{lin} using propagated centers $g(z_i) \in g(\Omega)$, i.e.

$$\exists z_1, .., z_d \in \Omega, \alpha_1, .., \alpha_d \in \mathbb{R}^b \ \forall x \in \Omega$$
$$\sum_{j=1}^{d} g^{(j)}(x) \cdot \beta_j = \sum_{i=1}^{d} k(g(x), g(z_i)) \cdot \alpha_i.$$

The proof is given in Section A.1. The second class of kernels used for the construction of the SDKNs is given by single-dimensional kernels $k_s : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{d\times d}$, where the index s refers to *single*. These kernels are composed of up to d different strictly positive definite kernels $k^{(1)}, ..., k^{(d)} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$:

$$\begin{aligned} k_s : \mathbb{R}^d \times \mathbb{R}^d &\to \mathbb{R}^{d\times d}, \\ (x, y) &\mapsto \operatorname{diag}(k^{(1)}(x^{(1)}, z^{(1)}), ..., k^{(d)}(x^{(d)}, z^{(d)})). \end{aligned} \tag{4.3}$$

The notation $x^{(i)}, z^{(i)}$ refers to the i-th coordinate entry of the corresponding vectors x respective $z \in \mathbb{R}^d$. We mostly focus on the case where $k^{(1)}, ..., k^{(d)}$ are one-dimensional radial basis function kernels and indeed we mostly use the case $k^{(1)} = ... = k^{(d)}$ for simplicity.

We remark that both the class of matrix-valued linear kernels and the class of single dimensional kernels are positive definite kernels, but in general not strictly positive definite kernels, see [120, Section 3.2.3] for more details.

For constructing the SDKNs, we use an odd number L of layers and proceed by using the linear kernel for all odd layers and single-dimensional kernels for all even layers. The input dimension d_1 and output dimension d_L of the SDKN are given by the problem at hand, and the intermediate dimensions $d_2, ..., d_{L-1}$ remain free to choose. For convenience, we usually simply use the same values for all intermediate layers, i.e. $d_2 = ... = d_{L-1}$. More details on commonly used values can be found in Section 5.1, where the SDKNs are applied for predicting turbulence closure terms.

An exemplary setup is visualized in Figure 4.1, where also similarities with neural networks are indicated:

1. Due to Proposition 23 and Proposition 24, the odd layers using the linear kernel essentially realize linear mappings as in neural networks. Thus, instead of optimizing the redundant parameters from the coefficients of the linear kernel expansion, we directly optimize the parameters of the corresponding matrices.

2. Due to Eq. (4.3), the mappings of the even layers are given by

$$f_l^{(j)}(x) = \sum_{i=1}^{N} \alpha_{l,i}^{(j)} k_j(x^{(j)}, F_{l-1}(z_i)^{(j)})$$

with $\alpha_{l,i}^{(j)} \in \mathbb{R}$. As all these parameters $\left(\alpha_{l,i}^{(j)}\right)_{i,j,l}$ can be optimized, the functions $f_l^{(j)}$ for $l = 2, 4, ...,\ j = 1, .., d_l$ can be interpreted as optimizable activation functions. This natural extension of fixed activation function as in neural networks is a benefit

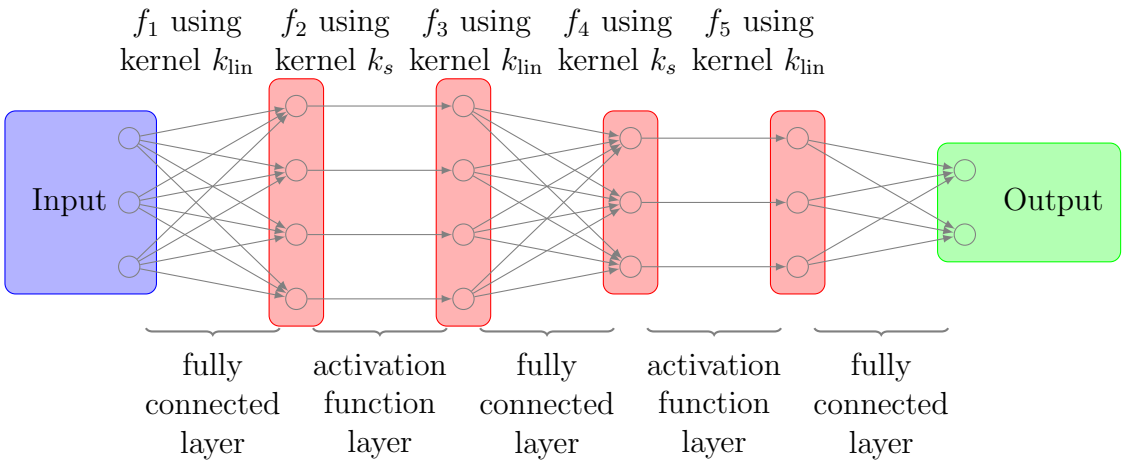

Figure 4.1: Exemplary visualization of a Structured Deep Kernel Network (SDKN) with 5 layers and input dimension 3 and output dimension 2. The braces below the layers indicate the similarities to neural networks. Figure taken from [116].

of the proposed approach, as it would also be possible to simply initialize but then *not* optimize those parameters.
In order to enhance sparsity and allow for a quicker evaluation, we do not optimize all the occurring parameters: For the even layers we consider only a subset of the training points, i.e. $n \ll N$ training points (called as before *centers*) of the whole data set X_N. Then, we optimize only the corresponding coefficients and leave the remaining ones constant to zero throughout the optimization of the coefficients:

$$f_l^{(j)}(x) = \sum_{i=1}^{M} \alpha_{l,i}^{(j)} k_j(x^{(j)}, F_{l-1}(z_i)^{(j)}). \tag{4.4}$$

This gives a tremendous speed-up of the optimization (in terms of computation time), but does neither harm the practical performance nor the theoretical approximation properties.

Finally, according to Eq. (2.35) and (2.36) the SDKN model using $M \ll N$ centers has the form

$$s_{\text{SDKN}}(\cdot) = \sum_{i=1}^{M} \mathcal{K}_L(\cdot, x_i)\alpha_i \tag{4.5}$$

with the deep kernel

$$k_{\text{SDKN}}(x, z) = K_L(f_{L-1} \circ .. \circ f_1(x), f_{L-1} \circ .. \circ f_1(z)). \tag{4.6}$$

The mappings f_l, $l = 1, ..., L-1$ are constructed using the matrix-valued linear kernel (for the odd layers, i.e. l odd) respective the single-dimensional kernels (for the even layers, i.e. l even) as elaborated above.

4.2.3 Optimization

For the optimization of the SDKN models constructed in Section 4.2.2, any loss and regularization according to Theorem 8 can be considered:

$$J(f_1, .., f_L) := \sum_{i=1}^{N} \mathcal{L}(y_i, f_L \circ .. \circ f_1(x_i)) + \sum_{l=1}^{L} \Theta_l(\|f_l\|_{\mathcal{H}_l^2}) \tag{4.7}$$

Therefore, it is pretty similar to the cost functional of Eq. (2.34), which is used for the optimization of neural networks, and thus one can leverage the optimization techniques, which are used for optimizing neural networks: In particular, the use of the Adam optimizer, which is "an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments" [57], showed good performance. Further, common state-of-the-art deep learning techniques, like random initialization that breaks symmetries, early stopping and gradient clipping [42] can be employed due to the structured setup of the SDKNs that is related to neural networks. The numerical experiments, see e.g. Section 5.1, show that the SDKN setup combined with these optimization techniques taken from the neural network / deep learning community provide sufficiently accurate models.

For the optimization of the SDKN model

$$s_{\text{SDKN}}(\cdot) = \sum_{i=1}^{M} \mathcal{K}_L(x, x_i)\alpha_i,$$

all the coefficients $\alpha_{l,i}^{(j)}$, $i = 1, ..., M$, $l = 1, ..., L$, of the (inner) layers (see Eq. (4.4)) are adjusted to minimize the cost functional J from Eq. (4.7). We remark that in contrast to Eq. (2.35), only $M \ll N$ data points were used as centers, which were usually picked randomly from the input training data set. The initialization of the coefficients $\alpha_{l,i}^{(j)}$ was done according to a normal distribution with a mean value above zero and a standard deviation of one. Experimentally, not much benefit was found by modifying the initialization or choice of centers. Due to this optimization according to the cost functional J, the SDKN model accurately reflects the input-output relation between the input data X_n and corresponding output data y_n. However one cannot expect the deep kernel

$$k_{\text{SDKN}}(x, z) = K_L(f_{L-1} \circ .. \circ f_1(x), f_{L-1} \circ .. \circ f_1(z)),$$

(see Eq. (2.36) and Eq. (4.6)) of the SDKN model to have special properties. In fact, it turns out that k_{SDKN} usually does not even generalize to new centers $\{\tilde{x}_i\}_{i=1}^{M}$: An approach as

$$\tilde{s}_{\text{SDKN}}(x) = \sum_{i=1}^{M} \mathcal{K}_L(\cdot, \tilde{x}_i)\alpha_i$$

with different centers $\{\tilde{x}_i\}_{i=1}^{M} \neq \{x_i\}_{i=1}^{M}$ does not yield a good performance, if only the final coefficients α_i, $i = 1, ..., M$ are tuned, but not the kernel $\mathcal{K}_L$ anymore. This is likely due to the used cost functional for the optimization. For this a different optimization idea is used in the two-layered approach introduced in Section 4.3. More details on this are given in Section 4.4, where the difference between the SDKN and the two-layered kernels approach are discussed.

4.2.4 Approximation properties of SDKNs

This section discusses the theoretical approximation properties of the proposed SDKN approach. For a detailed discussion we point to [120] and only summarize and comment on the main statements here. The proofs of the Theorems 26, 28 and 30 can be found in Section A.2, Section A.3 and Section A.4.

In the following, $\mathcal{F}_{L,w,M}$ denotes the class of functions that can be realized by any SDKN using $L \in \mathbb{N}$ single-dimensional kernel layers, a maximal dimension (width) of $w := \max(d_0, .., d_{2L+1}) \in \mathbb{N}$ and M centers. Like this, the total number of layers is $2L+1$. Furthermore we recall

$$\operatorname{dist}(\varphi, \mathcal{F}) := \inf_{f \in \mathcal{F}} \|\varphi - f\|_{L^\infty(\Omega)} \quad \text{for } \varphi \in C(\Omega)$$

for some $\mathcal{F} \subset C(\Omega)$.

- As a first limit case, the dependence on the number of centers n is discussed. The assumptions are collected in the following, and Theorem 26 shows a universal approximation property in the number of centers:

  Assumption 25 (Unbounded number of centers case)**.**

  1. *The kernels k_2, k_4 used in the single-dimensional mappings are universal kernels, i.e. their RKHSs are dense in the space of continuous functions.*
  2. *The centers can be chosen arbitrarily within $[0,1]^d$ and their number is unlimited (unbounded amount of centers).*

  Theorem 26 (Universal approx. for unbounded number of centers)**.** *Let $\Omega = [0,1]^d$. Consider an arbitrary continuous function $f : \Omega \to \mathbb{R}$. Then, it is possible under the Assumptions 25 to approximate this function f to arbitrary accuracy using an SDKN of finite width $(2d+1)d$ and finite depth $L = 2$, i.e.*

  $$\forall d \in \mathbb{N} \quad \lim_{M \to \infty} \operatorname{dist}(f, \mathcal{F}_{2,(2d+1)d,M}(\Omega)) = 0.$$

- A second limit case is provided by the width of the SDKN. Again, we collect the assumptions and conclude with the statement in Theorem 28, whereby we note that the second assumption is pretty trivial.

  Assumption 27 (Unbounded width case)**.** *1. The radial basis function φ of the single-dimensional kernel k_2 of the mapping f_2 (see e.g. Eq. (4.4)) needs to satisfy $\overline{\operatorname{span}\{\varphi(ax) | a > 0\}} = C([0,1])$ for a given $\varphi : \mathbb{R}_+ \to \mathbb{R}$.*

  2. At least 2 different centers $z_1 \neq z_2$ are given.

  Theorem 28 (Universal approximation for unbounded width)**.** *Let $\Omega = [0,1]^d$. Consider an arbitrary continuous function $f : \Omega \to \mathbb{R}$. Then, it is possible under the Assumptions 27 to approximate this function f to arbitrary accuracy using an SDKN of depth $L = 1$ and sufficient width, i.e.*

  $$\forall d \in \mathbb{N} \quad \lim_{d_1 = d_2 \to \infty} \operatorname{dist}(f, \mathcal{F}_{1,d_1,2}(\Omega)) = 0.$$

- The third and probably most important limit case is given in the depth of the SDKN:

  Assumption 29 (Unbounded depth case)**.** *1. The kernels of the single activation function layers satisfy the requirements of Eq.* (A.4)*.*

  2. There are 3 *centers* $z_1, z_2, z_3 \in \mathbb{R}^d_{\geq 0}$ *given such that* $z_1^{(j)}, z_2^{(j)}, z_3^{(j)}$ *are pairwise distinct for* $j = 1, .., d$*.*

  Theorem 30 (Universal approximation for unbounded depth)**.** *Let* $\Omega = [0,1]^d$*. Consider an arbitrary continuous function* $f : \Omega \to \mathbb{R}$*. Then it is possible under the Assumptions 29 to approximate this function* f *to arbitrary accuracy using an SDKN of width* $w = d + 8$ *and* 3 *centers, i.e.*

$$\forall d \in \mathbb{N} \lim_{L \to \infty} \operatorname{dist}(f, \mathcal{F}_{L,d+8,3}(\Omega))) = 0.$$

Theorem 26 and Theorem 30 discuss the cases of unbounded width and depth, which were also researched for neural networks, for which also universal approximation statements can be derived. Additionally, due to the kernel setup of the SDKNs, Theorem 26 also derives a statement for unbounded number of centers.

Of particular interest is the statement of Theorem 30: Here it is possible to improve related constructions for neural networks by making use of kernel specific approximation properties, in particular of the so-called "flat limit" of RBF kernels. For more details we refer to [120, Section 4.3].

We remark that the theorems were formulated for $\Omega = [0,1]^d$, but generalizations to other domains are also possible. In the case of bounded domains, one can always consider a bounding box $[-b, b]^d$ for $b > 0$ large enough and thus rely on the analysis for $[0,1]^d$.

4.2.5 Connections to neural networks

Instead of using the single-dimensional kernels as before, one could also make use of primal defined activation function kernels: Given an activation function $\sigma : \mathbb{R} \to \mathbb{R}, x \mapsto \sigma(x)$ such as $\sigma(x) = \mathrm{ReLU}(x)$, one can consider the positive definite matrix-valued kernel defined via the feature map given by the elementwise acting activation function σ as

$$\begin{aligned} k: \ \mathbb{R}^d \times \mathbb{R}^d &\to \mathbb{R}^{d \times d} \\ (x, y) &\mapsto \langle \sigma(x), \sigma(y) \rangle_{\mathbb{R}^d} \cdot I_d. \end{aligned} \tag{4.8}$$

Like this the kernel mapping (in the component $\mu \in \{1, ..., d\}$) has the following form:

$$\begin{aligned} s^{(\mu)}(x) &= \sum_{i=1}^{M} (k(x, z_i)\alpha_i)_\mu \equiv \sum_{i=1}^{M} \alpha_{\mu,i} \langle \sigma(x), \sigma(z_i) \rangle_{\mathbb{R}^d} \\ &= \sum_{i=1}^{M} \alpha_{\mu,i} \sum_{j=1}^{d} \sigma(x^{(j)}) \sigma(z_i^{(j)}). \end{aligned}$$

Combined with a subsequent linear mapping (see Proposition 23) due to the linear kernel in the odd layers we obtain

$$\sum_{\mu=1}^{d} \beta_\mu s^{(\mu)}(\cdot) = \sum_{j=1}^{d} \sigma(x^{(j)}) \cdot \sum_{\mu=1}^{d} \beta_\mu \sum_{i=1}^{M} \alpha_{\mu,i} \sigma(z_i^{(j)}).$$

This shows that the SDKN setup combined with the primal defined kernels from Eq. (4.8) yields standard neural networks (if biases are additionaly incorporated). It remains a future research direction to investigate the implications of the deep kernel representer theorem to neural networks via the connection established with help of the primal defined activation function kernels from Eq. (4.8).

4.2.6 Discussion and comments

To the best of the author's knowledge, the SDKN approach was the first deep kernel approach, which made explicitly use of the deep kernel representer theorem, see Theorem 8. Due to this, the SDKN approach is inherently optimal in the sense of the representer theorem. By using only a subset of the centers for the single-dimensional kernel layers and leveraging Proposition 23 for the linear kernel layers, it was furthermore possible to derive a sparse representation of the SDKNs which allowed for an efficient optimization. As already remarked before in Section 4.3.2, the resulting deep kernel of the SDKN from Eq. (4.6) is actually not a useful kernel itself, in the sense that it is a proper data adapted kernel and can be used in conjunction with other kernel algorithms on the same data set on which the SDKN was trained on.

Thus, for **future research** the following points are of interest:

- (SDKN-kernel): The SDKNs are capable of achieving state-of-the-art performance in several applications, especially dealing with large datasets, see e.g. Section 5.1 or [116, 117]. However the SDKN-kernel from Eq. (4.6) does not generalize well so far. Thus, an open question is how to modify the SDKNs or the applied optimization procedure in order to derive not only a powerful model (the SDKN), but also a deep kernel that generalizes well (SDKN-kernel). Some comments on this point can be found in Section 4.4, after the introduction of the two-layered kernels in the next Section 4.3.

- (Connection to neural networks): Another relevant line of future research related to the SDKN approach is about its connection to neural networks as explained in Section 4.2.5: By picking primal defined kernels using pointwise activation functions as feature maps, it is possible to exactly represent neural networks. Thus, the implications of the deep kernel representer theorem for standard neural networks could be of interest here, possibly giving further insights into the training and generalization of neural networks.

4.3 Two-layered kernels (2L)

This subsection is devoted to the introduction of two-layered kernels and, in particular, the 2L-VKOGA algorithm. In contrast to the SDKN model introduced in the previous

subsection, this approach does not only give an accurate overall kernel model, but in particular also a suitable deep kernel, which can be used in conjunction with other kernel algorithms – e.g. greedy algorithms. Most of the content of this subsection was covered in [118] with some extensions in [116].

We start by motivating and constructing the two-layered kernels in Section 4.3.1, discussing their optimization and analytical as well as computational properties in Section 4.3.2, Section 4.3.3 and Section 4.3.4 and conclude by commenting on possible extensions in Section 4.3.5.

4.3.1 Construction of two-layered kernels

Radial basis function kernels as in Eq. (2.1) can be chosen depending on a shape parameter. While analytic properties like convergence rates or stability estimates asymptotically usually do not depend on the choice of that hyperparameter, the performance in numerical experiments using a finite data set crucially depends on its choice. Therefore several ways for choosing this shape hyperparameter have been introduced in the literature, as discussed in Section 4.1. While heuristic choices are a suitable way to go for a first try, cross-validation usually provides a more principled way and frequently gives better results. However, this cross-validation can be computational expensive, see also Section 4.3.3 for a discussion.

For this reason two-layered kernels were introduced in [118], which can be seen as a generalization of this hyperparameter problem: By generalizing the single shape hyperparameter to a $d \times d$ matrix of d^2 hyperparameters, we allow for arbitrary rotations and anisotropic scaling of the input space for the kernel. For this we consider the $d \times d$ matrix A_θ

$$A_\theta = \begin{pmatrix} \theta_1^1 & \dots & \theta_1^d \\ \vdots & \ddots & \vdots \\ \theta_d^1 & \dots & \theta_d^d \end{pmatrix},$$

and consider the corresponding kernel

$$\begin{aligned} k_\theta : \ \mathbb{R}^d \times \mathbb{R}^d &\to \mathbb{R} \\ (x, y) &\mapsto k(A_\theta x, A_\theta y). \end{aligned} \tag{4.9}$$

for some RBF base kernel $k : \mathbb{R}^b \times \mathbb{R}^b \to \mathbb{R}$. From here it is evident that this approach is a generalization: For the choice $A_\theta = \varepsilon \cdot I_d$ we obtain the standard shape parameter scaled kernel from Eq. (2.1). Using additionally $\varepsilon \to 0$ yields the flat limit of kernels, which was also mentioned in Section 4.2.4. For $A_\theta = \operatorname{diag}(\varepsilon_1, ..., \varepsilon_d)$ we obtain again anisotropic kernels.

Recalling Proposition 23, we saw that linear mappings like $x \mapsto A_\theta x$ can be realized with the help of a matrix-valued linear kernel. Thus, the concatenation of this linear map with an RBF kernel k as in Eq. (4.9) fits naturally into the framework of the deep kernel representer Theorem 8. This connection was also formalized in [118, Corollary 3.1], which we restate here and also recall the proof:

Corollary 31. *Consider* $\{x_1, ..., x_N\} \subset \mathbb{R}^d$ *such that the data matrix* $[x_1, ..., x_N] \in \mathbb{R}^{d \times N}$ *has rank* d*. Then, the kernel* k_θ *from Eq.* (4.9) *is an instance of a two-layered kernel according to the deep kernel representer Theorem 8.*

Proof. Due to the assumption on the rank d of the data matrix, Proposition 23 can be leveraged to represent the linear mapping $x \mapsto A_\theta x$ as a kernel model using the matrix-valued linear kernel, i.e. we obtain

$$\sum_{j=1}^{N} k_{\text{lin}}(x, x_j)\alpha_j^{(1)} = A_\theta x.$$

Thus, using this mapping as a first layer mapping f_1, the general two-layered kernel $k(f_1(x), f_1(z))$ (i.e. Eq. (2.36) for $L = 2$) specializes to

$$\mathcal{K}_2(x, y) = k(f_1(x), f_1(y)) = k(A_\theta x, A_\theta y),$$

i.e. the hyperparameter tunable kernel k_θ from Eq. (4.9) is indeed an instance of a two-layered kernel. □

The assumption on the rank d of the data matrix $[x_1, ..., x_N] \in \mathbb{R}^{d \times N}$ in Corollary 31 deserves some comments: Assume that the data matrix has rank less than d. In this case the data can be embedded in some lower dimensional space $\mathbb{R}^{d'}$ with $d' < d$, and there is no data along the orthogonal directions. In this case there is no knowledge about the behavior of the underlying data generating mechanism in these orthogonal directions and it makes little sense to use a kernel which varies in those directions. Indeed, using a proper regularization term during the optimization of the kernel, this can be automatically detected and achieved, see Section 4.3.2 for more details.

Independent of the rank of the data matrix, the purpose of optimizing such a two-layered kernel is to incorporate information from the target values $[y_1, ..., y_N] \in \mathbb{R}^N$ already into the kernel k_θ: If the target values do not change at all in some direction, then also the kernel k_θ should not change in this direction. Similar behavior can be expected, when the target values change e.g. only in a slow and smooth way. This will be achieved with a proper optimization strategy, see the next subsection.

We remark that the first layer with its equivalent matrix A_θ can either be interpreted as a modification of the kernel itself, or as a transformation of the input space. In the following, we make use of both interpretations.

A visualization of the two-layered kernel approach is given in Figure 4.2.

4.3.2 Optimization of two-layered kernels

The optimization of the two-layered kernel from Eq. (4.9) is equal to the optimization of the matrix $A_\theta \in \mathbb{R}^{d \times d}$. A direct global computation of an optimal matrix A_θ seems to be infeasible, because this would involve all the data points (X_N, Y_N), which yields huge matrices:

$$\min_{A_\theta \in \mathbb{R}^{d \times d}} \ell(X_N, Y_N, A_\theta). \tag{4.10}$$

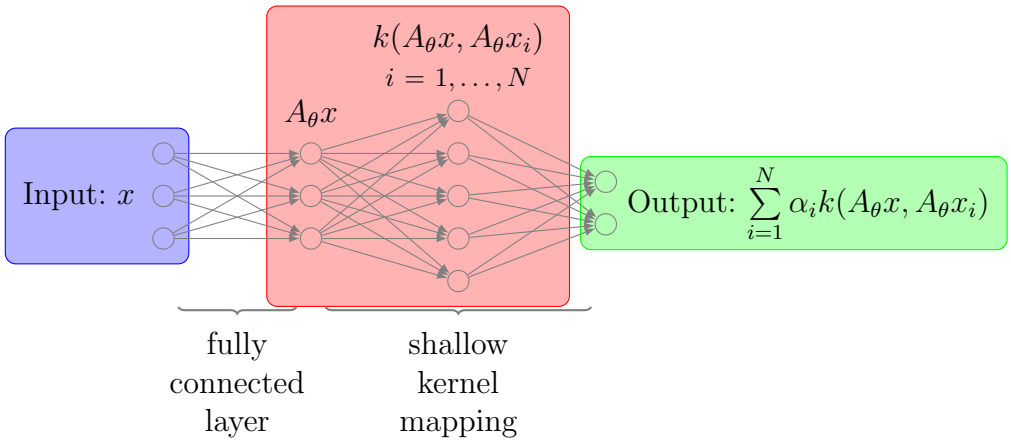

Figure 4.2: Exemplary visualization of a two-layered kernel model with $x \in \mathbb{R}^3$, $A_\theta \in \mathbb{R}^{3\times 3}$, $N = 5$, $x_i \in \mathbb{R}^3$ and $\alpha_i \in \mathbb{R}^2$ for $i = 1, \ldots, N$. Figure taken from [116] and modified.

Therefore we employ a gradient descent mini batch training: For the initialization we use the identity matrix for A_θ, i.e. $A_\theta = I_d$ and then start with the optimization: For this we choose a batch size $n_{\text{batch}} \ll N$ and then randomly draw disjoint mini batches $(X_{\text{batch}}, Y_{\text{batch}}) \subset (X_N, Y_N)$, on which a loss function is evaluated. As a loss function we pick the sum of m-fold cross-validation errors with $2 \leq m \leq n_{\text{batch}}$, which can be computed efficiently via an extension of Rippa's rule [69]. The case $m = n_{\text{batch}}$ corresponds to the leave-one-out-cross-validation:
Denote $X_{n_{\text{batch}}} = \{x_1, ..., x_{n_{\text{batch}}}\}$, $y_{n_{\text{batch}}} = \{y_1, ..., y_{n_{\text{batch}}}\}$ and assume for simplicity that $n_{\text{batch}}/m =: p \in \mathbb{N}$. For every fold, i.e. every $j = 1, ..., m$:

- Let $r := r^{(j)} := (r_1, ..., r_p)^\top$ be the vector of pairwise distinct validation indices.
- Then the validation set is given by $V_p := V_p^{(j)} := \{x_{r_i}, i = 1, ..., p\} \subset X_{n_{\text{batch}}}$ and we are interested in the vector of residuals $e_{\text{res}}^{(j)} := (e_1, ..., e_p)$ given as $e_i := s_{X_{n_{\text{batch}}} \setminus V_p}(x_{r_i}) - y_{r_i}$ for $i = 1, ..., p$. Hereby $s_{X_{n_{\text{batch}}} \setminus V_p}$ is the kernel approximant based on the training set $X_{n_{\text{batch}}} \setminus V_p \subset X_{n_{\text{batch}}}$.
  [70, Theorem 1] then showed, that $e_{\text{res}}^{(j)}$ can be computed efficiently as the solution of

$$(A_{X_{\text{batch}}})_{V_p}^{-1} e_{\text{res}}^{(j)} = (y_{r_1}, ..., y_{r_p})^\top, \tag{4.11}$$

  whereby $(A_{X_{\text{batch}}})_{V_p}^{-1}$ is the $p \times p$ submatrix (defined by the corresponding indices) of the inverse kernel matrix $A_{X_{\text{batch}}}$.

The loss of interest for the optimization in every step of the mini batch gradient descent is then given as

$$\ell_p(X_{n_{\text{batch}}}, Y_{n_{\text{batch}}}, A_\theta) = \sum_{j=1}^{p} \|e_{\text{res}}^{(j)}\|_2^2. \tag{4.12}$$

Depending on the way the kernel approximants $s_{X_{n_{\text{batch}}} \setminus V_p}$ are computed, further hyperparameter can come into play in Eq. (4.10). Especially it is advisable to compute the

kernel approximant in a regularized way to stabilize the optimization procedure from a numerical point of view. This can be done by making use of the regularized kernel matrix $A_{X_{\text{batch}}} + \lambda I_{n_{\text{batch}}}$ for some $\lambda > 0$ instead of the pure kernel matrix $A_{X_{\text{batch}}}$, cf. Eq. (2.8). Therefore, we denote the loss from Eq. (4.12) also as $\ell_{p,\lambda}$. Furthermore, we remark that it is in principle also possible to consider other norms in Eq. (4.12), however the ℓ^2 norm is convenient.

The overall algorithm is depicted in Algorithm 1, which was taken from [118] with only minor modifications. The depicted GET_BATCH method draws an exhausting collection of batches, though also random drawings should give comparable results. The EARLY_STOPPING method is optional, and we point the interested reader to [42]. We remark that for the numerical experiments in Chapter 5 the more sophisticated Adam optimizer [57] is used instead of a plain stochastic gradient descent update.
The algorithm is intrinsically supervised because in the key computation step in Eq. (4.11) for building the final loss of Eq. (4.12), the target values y are explicitly used. Here, it is emphasized that the incorporation of the target values is crucial for the subsequent performance in the numerical applications. Especially in plenty of applications for surrogate modeling, the input space is randomly or uniformly sampled and the corresponding target values are subsequently computed. In this view, one cannot expect to obtain any useful information solely based on the input values without taking into account the corresponding output values.

While [118] introduced the model originally only for scalar valued outputs, an extension to the vectorial case is feasible and was done in [116] by using a separable matrix-valued kernel. In this case, the overall loss (for the multivariate output case) is given by the ℓ^2 norm over the individual output dimensions, i.e. in the framework of Eq. (4.12) the residuals $e_{\text{res}}^{(j)}$ are now matrix-valued instead of vector valued.

Concerning the computational complexity of the kernel optimization in contrast to straightforward cross-validation, we note: The gradient descent kernel optimization with a maximum number n_{epoch} of epochs and a given batch size n_{batch} requires

$$\mathcal{O}(n_{\text{epoch}} \cdot N/n_{\text{batch}} \cdot n_{\text{batch}}^3) = \mathcal{O}(n_{\text{epoch}} \cdot N \cdot n_{\text{batch}}^2),$$

operations for calculating the losses from Eq. (4.11). This is in contrast to a full cross-validation of the parameters: Assuming n_{CV} runs per parameter, the number of required loss evaluations scales as $n_{\text{CV}}^{d^2}$ for a full validation of the matrix A_θ or n_{CV}^d if only the diagonal values of A_θ are validated. Both versions clearly suffer from the curse of dimensionality in contrast to the gradient descent optimization of the matrix A_θ.

4.3.3 Analysis and investigation of two-layered kernels

In this subsection, we quickly recall the most important statements from the theoretical analysis of the two-layered kernels from [118, Section 3.2].

- Symmetry of A_θ: Due to using an RBF kernel k in the second layer, it is enough to consider symmetric matrices A_θ: Given the singular value decomposition

$$A_\theta = U\Sigma V^\top \Leftrightarrow A_\theta v_i = \sigma_i u_i \quad \text{for } i = 1, ..., d$$

Algorithm 1: SGD optimization of the kernel $k_\theta(x, y) \equiv k(A_\theta x, A_\theta y)$.

Input : Data (X_N, Y_N), base kernel k, m-fold parameter m, regularization parameter λ
learning rate μ, batch size n_{batch}, number epochs $\max_{\text{epoch}}$
Result: Optimized matrix A_θ
$p := \lfloor n_{\text{batch}}/m \rfloor$, $\max_{\text{iter}} := \lfloor N/n_{\text{batch}} \rfloor$
$A_\theta \leftarrow \text{diag}(1, ..., 1)$; `/* Initialization of` A_θ `*/`

for $n_{\text{epoch}} = 1, ..., \max_{\text{epoch}}$ **do**
  $L_{\text{epoch}} \leftarrow 0$;
  SHUFFLE (X_N, Y_N);
  for $n_{\text{iter}} = 1, ..., \max_{\text{iter}}$ **do**
    $(X_{\text{batch}}, Y_{\text{batch}}) \leftarrow$ GET_BATCH$((X_N, Y_N))$;
    $L = \ell_{p,\lambda}(X_{\text{batch}}, Y_{\text{batch}}, A_\theta)$; `/* Using Eq. (4.12) */`
    $A_\theta \leftarrow A_\theta - \mu \cdot \frac{\partial L}{\partial A_\theta}$; `/* Gradient descent update */`
    $L_{\text{epoch}} \leftarrow L_{\text{epoch}} + L$
  end
  EARLY_STOPPING(L_{epoch})
end

with $\Sigma = \text{diag}(\sigma_1, ..., \sigma_d)$ and u_i and v_i for $i = 1, .., d$ being the column vectors of U, V, i.e. the left and right singular vectors. Then we have

$$\|A_\theta(x - \tilde{x})\|^2_{\mathbb{R}^d} = \sum_{i=1}^{d} (\langle x, v_i \rangle_{\mathbb{R}^d} - \langle \tilde{x}, v_i \rangle_{\mathbb{R}^d})^2 \sigma_i^2,$$

because the left singular vectors $u_i, i = 1, ..., d$ vanish due to taking the norm. This allows to choose $U := V$, such that the resulting matrix A_θ is symmetric without changing its singular values nor its right singular vectors.

- Singular value behavior of A_θ: In order to assess whether the optimized kernel is likely beneficial in comparison to the standard kernel (i.e. using the initial value $A_\theta = I_d$), the decay of the singular values of the matrix A_θ can be investigated: We assume the singular values $\sigma_i(A_\theta)$ to be ordered from the largest to the smallest one according to their absolute value. Then we consider the cumulative energy

$$n \mapsto \frac{\sum_{i=1}^{n} |\sigma_i(A_\theta)|}{\sum_{i=1}^{d} |\sigma_i(A_\theta)|}, \quad 1 \leq n \leq d, \tag{4.13}$$

  which measures how much energy is captured in the top singular values. The faster the cumulative energy approaches the value one, the more energy is clustered in a few singular values, while the remaining singular values need to be comparably small. The quantity Eq. (4.13) is visualized depending on the ratio n/d of considered eigenvalues (x-axis) in Figure 4.3:

  One can see, that for the considered test datasets (for more details see the experiments in Section 4.3.4), the ratio approaches the value one more or less quickly. In

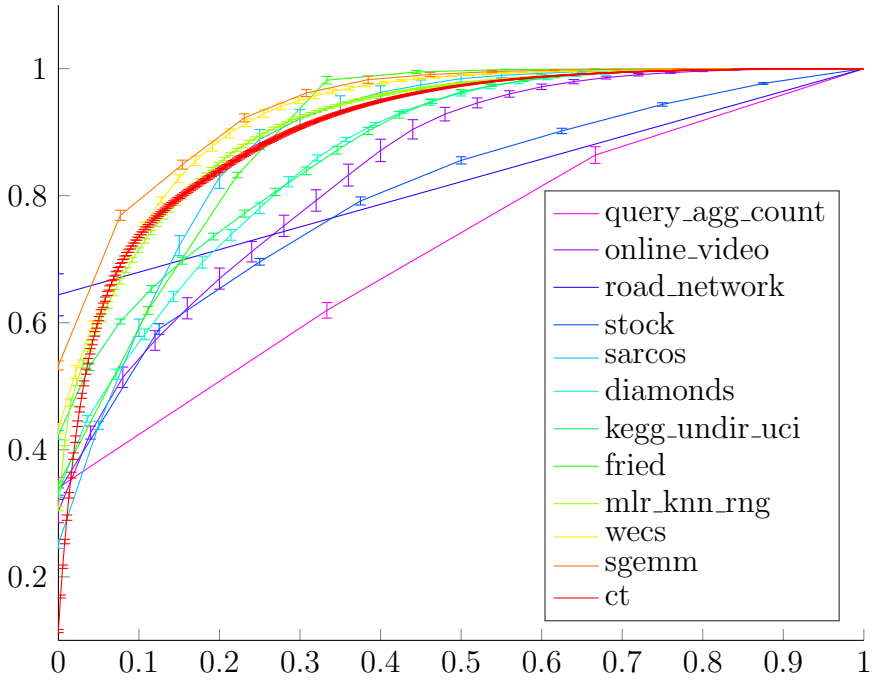

Figure 4.3: Visualization of the ratio $\sum_{i=1}^{n} |\sigma_i(A_\theta)| / \sum_{i=1}^{d} |\sigma_i(A_\theta)|$ (mean and standard deviation over 5 reruns) (y-axis) over the ratio n/d of considered singular values (x-axis) using $m = n_{\text{batch}} = 64$. Figure taken from [118].

view of Figure 4.6 and Figure 4.7 (elaborated in Section 4.3.4), one can see that the kernel optimization in conjunction with a subsequent use of greedy algorithms yields better results, the faster the ratio Eq. (4.13) approaches the value one. Thus, the investigation of Eq. (4.13) can serve as an a priori assessment whether it makes sense to use the optimized kernel for running a subsequent kernel approximation algorithm (which is likely more costly than the kernel optimization, see comments in Section 4.3.4 and run times in Table 4.2).

- Stability of the optimization: The optimization provided by Algorithm 1 was investigated numerically in [118, Section 4.3] and found to be sufficiently stable, in the sense that the final optimized kernel is indeed data-adapted and more or less always the same independent of the randomness incurred due to the stochastic optimization or the choice of the m-fold parameter m: For this experimental investigation, the three exemplary regression datasets mlr_knn_rng, fried and kegg_undir_uci [36, 51, 104] were considered, see Table 4.1 for information about these data sets. For each of these datasets the two-layered kernel optimization was performed 5 times for the m-fold values $m \in \{64, 32, 16, 8, 4\}$ using a batch size of 64. In order to analyze the resulting kernel k_θ, the first layer matrix A_θ is analyzed with a singular value decomposition. Figure 4.4 visualizes the singular values as well as the overlap of the corresponding right singular vectors. To be precise, for the second quantity, the principal angle between subspaces spanned by right singular vectors corresponding to the largest $n \in \{1, ..., 15\}$ singular values is plotted, whereby we compare the subspaces corresponding to the $m = 64$ optimization with $m \in \{32, 16, 8, 4\}$.

Short name	Number of examples	Number of features
fried	40768	10
sarcos	44484	21
ct	53500	379
diamonds	53940	29
stock	59049	9
kegg_undir_uci	64608	27
online_video	68784	26
wecs	72000	48
mlr_knn_rng	111753	132
query_agg_count	200000	4
sgemm	241600	14
road_network	434874	2

Table 4.1: Overview on the machine learning data sets which were used.

For a precise definition of the principal angle between subspaces, the interested reader is pointed to [58, Section 1] – for the understanding it is sufficient to recognize it as a measure of the overlap of the subspaces. The left column within Figure 4.4 shows, that the distribution of the singular values always follows the same behavior – independent of the actually chosen value m for the m-fold cross-validation error. The right column shows, that the right singular vectors to the top singular values are quite aligned, because the corresponding principal angles are small in the beginning. With growing dimension of the considered subspaces also the principal angles increase, which makes sense in view of the high dimensionality of the considered datasets (see Table 4.1). Like this we conclude that the features (given by the projection on the right singular vectors) identified by the different optimization criteria as well as their importance (given by the singular values) are pretty similar independent of the choice of hyperparameter m for the optimization, i.e. the features and their importance are data related.

4.3.4 Two-layered greedy algorithms: 2L-VKOGA

As motivated and elaborated before, the two-layered kernel k_θ can be used in conjunction with general kernel-based algorithms. The combination of two-layered kernel optimization with subsequent greedy kernel approximation, in particular together with the VKOGA algorithm [93], was labeled $2L\text{-}VKOGA$ in [118], whereby the actual treatment of the vectorial case was done in [116]. An implementation can be found at

`https://gitlab.mathematik.uni-stuttgart.de/pub/ians-anm/2l-vkoga`.

Usually the computational cost for the kernel optimization is significantly smaller than running the subsequent greedy algorithm, such that the kernel optimization does not bear much computational overhead: As analyzed in Section 4.3.2, the computational cost for the kernel optimization using (up to) n_{epoch} epochs is

$$\mathcal{O}(n_{\text{epoch}} \cdot N/n_{\text{batch}} \cdot n_{\text{batch}}^3) = \mathcal{O}(n_{\text{epoch}} \cdot N \cdot n_{\text{batch}}^2),$$

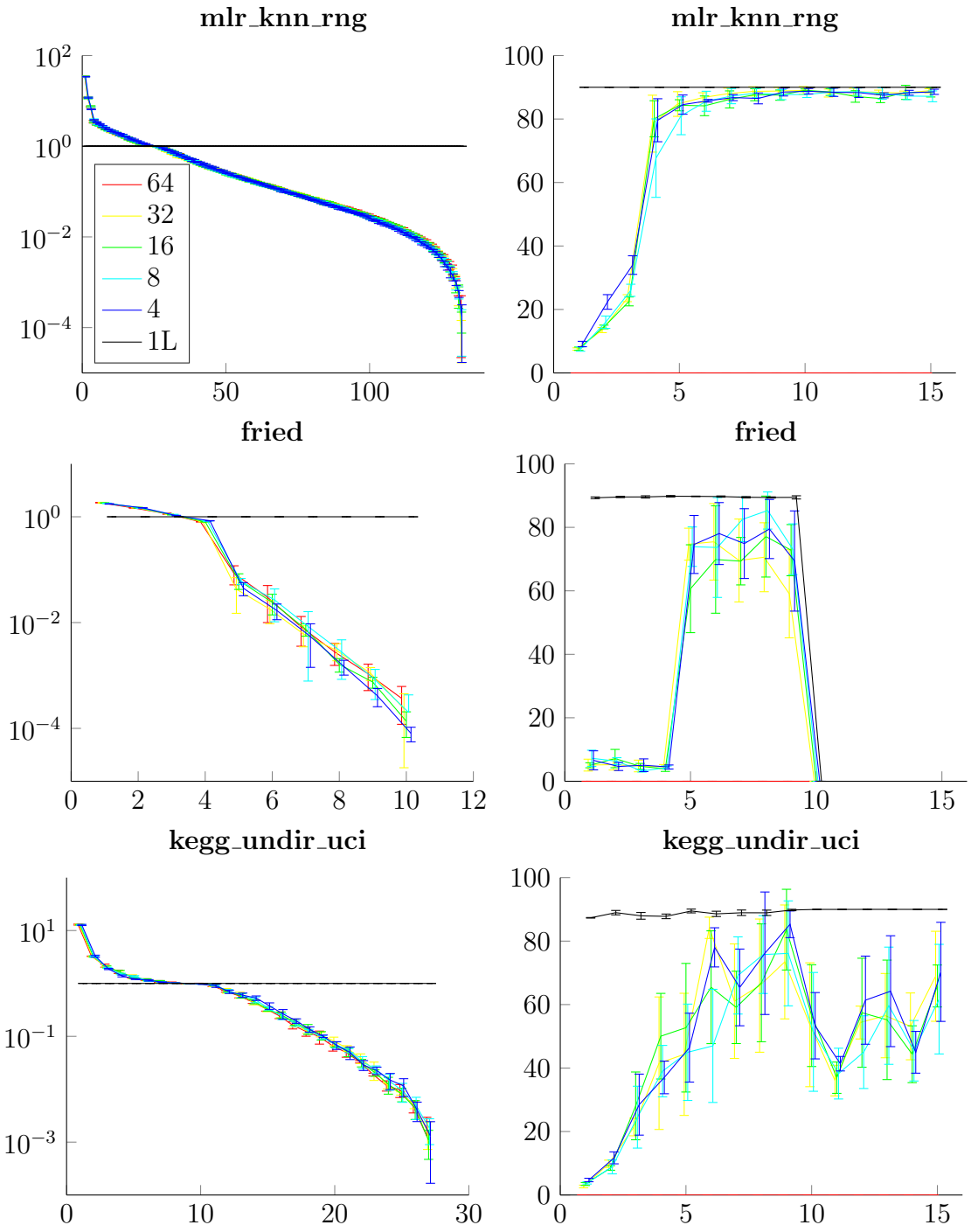

Figure 4.4: Analysis of the optimized matrices A_θ for three datasets. Left: Visualization of the singular values of the matrix A_θ. Right: Visualization of the principal angle (y-axis) between subspaces spanned by the first n right singular vectors (x-axis). We set the number of folds as $n_{\text{fold}} = 64$ as reference and compare to $n_{\text{fold}} \in \{64, 32, 16, 8, 4\}$ and no optimization (i.e. the classical "1L"-VKOGA). Error bars are used to visualize the mean and standard deviation for 5 reruns. Figure taken from [118].

Short name	Time for optimization [s]	Time for greedy [s]
fried	$13.90 \pm 3.51s$	$83.16 \pm 0.66s$
sarcos	$12.94 \pm 2.32s$	$92.71 \pm 0.43s$
ct	$64.66 \pm 5.79s$	$250.93 \pm 1.57s$
diamonds	$19.40 \pm 2.25s$	$113.61 \pm 0.74s$
stock	$16.48 \pm 2.20s$	$116.63 \pm 0.49s$
kegg_undir_uci	$25.48 \pm 4.29s$	$133.85 \pm 0.75s$
online_video	$20.66 \pm 1.94s$	$140.72 \pm 0.99s$
wecs	$28.92 \pm 3.65s$	$162.33 \pm 0.85s$
mlr_knn_rng	$95.57 \pm 12.90s$	$308.29 \pm 0.79s$
query_agg_count	$62.73 \pm 7.66s$	$351.29 \pm 0.98s$
sgemm	$90.66 \pm 4.41s$	$446.20 \pm 1.99s$
road_network	$133.23 \pm 8.91s$	$733.22 \pm 13.27s$

Table 4.2: Overview of time spent for kernel optimization and subsequent model computation via greedy (mean and standard deviation over 5 reruns).

and thus in particular independent of the VKOGA expansion size n_{vkoga}. In contrast, the computational cost of VKOGA scales even quadratically in the expansion size n_{vkoga}:

$$\mathcal{O}(n_{\text{vkoga}}^2 \cdot N).$$

Typical values, which were also used in the numerical experiments (see below and Figure 4.6, 4.7) are $n_{\text{batch}} = 64, n_{\text{epoch}} = 10, n_{\text{vkoga}} = 1000$. These values typically yield that the kernel optimization before running VKOGA is only a small computational overhead. This is also reflected in the timings of the numerical examples related to Figure 4.6, 4.7, see also Table 4.2.

For the convergence analysis of these two-layered greedy algorithms we recall [118, Theorem 3.2] and [118, Theorem 3.3]. The proofs are deferred to Chapter B. Both theorems are formulated in terms of fill distances which corresponds to the target data-independent P-greedy algorithm. By applying the analysis of Chapter 3, it is possible to also derive target data-dependent convergence rates for e.g. the f-greedy algorithm. As this is a straightforward application of the theory, it is not included explicitly here.

Theorem 32. *Consider an RBF kernel k that satisfies Eq. (2.2) with $\tau > d/2$ on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$. Consider $f \in \mathcal{H}_k(\Omega)$ and the kernel interpolant s_{X_n} using the two-layered kernel k_θ with $A_\theta \in \mathbb{R}^{d\times d}$ and $\operatorname{rank}(A_\theta) = d$.*
Then the following pointwise error estimate holds:

$$|f(x) - s_{X_n}(x)| \leq C h_{X_n,\Omega}^{\tau - d/2}.$$

For asymptotically uniformly distributed points $h_{X_n,\Omega} \asymp n^{-1/d}$ it then holds

$$|f(x) - s_{X_n}(x)| \leq C n^{1/2 - \tau/d}.$$

Theorem 33. *Consider a Matérn kernel k with $\tau > d/2$ on a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$. Consider $f \in \mathcal{H}_k(\Omega)$ and the kernel interpolant s_{X_n} using the two-layered kernel k_θ with $A_\theta \in \mathbb{R}^{d\times d}$ and $\operatorname{rank}(A_\theta) =: d_{\mathit{eff}} < d$.*

Assume that $f \in \mathcal{H}_k(\Omega)$ *is invariant along the subspace* $\mathrm{Null}(A_\theta) \subset \mathbb{R}^d$*, i.e.* $f(x) = f(x')$ *for any* $x, x' \in \Omega$ *with* $x - x' \in \mathrm{Null}(A_\theta)$*. For points* $X_n \subset \Omega$ *such that* $h_{A_\theta\Omega, A_\theta X_n} \asymp n^{-1/d_{\mathit{eff}}}$ *it then holds*

$$|f(x) - s_{X_n}(x)| \leq C n^{\frac{d}{2d_{\mathit{eff}}} - \frac{\tau}{d_{\mathit{eff}}}}.$$

Theorem 32 shows for the case $\mathrm{rank}(A_\theta) = d$ that the asymptotic convergence rate does not decrease when using the two-layered kernel compared to using a standard one-layered kernel (i.e. $A_\theta = I_d$). Theorem 33 shows for the case $\mathrm{rank}(A_\theta) < d$ that the asymptotic convergence rate is even improved, which is due to reducing the effective dimension of the problem. Although Theorem 32 does not give any advantage in terms of faster convergence rate over using the standard kernel k, the two-layered kernel still usually performs better in the preasymptotic regime.

From the theoretical point of view it remains unclear whether it is possible to obtain faster asymptotic convergence rates for target data-dependent greedy algorithms when using a two-layered kernel instead of a standard shape tuned kernel, for the case where all the singular values $s_i(A_\theta)$ of A_θ are bounded away from zero: In order to answer these questions, a more refined analysis of the f-greedy algorithm seems to be required.

In order to demonstrate the usefulness of the 2L-VKOGA approach and also elucidate Theorem 32 and Theorem 33, we recall two numerical examples [118, Section 4.1] and [118, Section 4.2]. First, the performance of the f-greedy algorithm using an optimized two-layered kernel (with a basic Matérn kernel $k(x, y) = \exp(-\|x - y\|/\sqrt{d})$ as base kernel) versus using standard kernels of different shape parameter (10 logarithmically equally spaced values in between 0.05 and 10) was investigated numerically in [118, Section 4.1]: The domain $\Omega = [0, 1]^d$ for $d \in \{5, 6, 7\}$ and the three test functions

$$f_d(x) = \begin{cases} \mathrm{e}^{-4\left(\sum_{j=1}^5 x^{(i)} - 0.5\right)^2}, & \text{for } d = 5, \\ \mathrm{e}^{-4\sum_{j=1}^6 (x^{(i)} - 0.5)^2} + 2|x^{(1)} - 0.5|, & \text{for } d = 6, \\ \mathrm{e}^{-\sum_{j=1}^7 (x^{(i)} - 0.5)^2} + \mathrm{e}^{-9\sum_{j=1}^2 (x^{(i)} - 0.3)^2}, & \text{for } d = 7. \end{cases}$$

were used. The resulting decay of the $\|\cdot\|_{L^\infty(\Omega)}$ training error (which was checked to coincide with a test error) for the approximation is depicted in Figure 4.5. There seem to be two different cases for the convergence behavior of the f-greedy algorithm using a two-layered kernel:

- Test function f_5 (top row in Figure 4.5) shows a faster convergence rate for f-greedy using the optimized two-layered kernel over using a standard shape parameter tuned kernel.
- Test function f_6, f_7 (middle and bottom row in Figure 4.5) show the same convergence rate for the optimized two-layered kernel as for the standard shape parameter tuned kernel. However, the occurring prefactor is way smaller, such that one obtains constant multiplicative benefit.

However, as stressed before, for numerical experiments also the preasymptotic range is of interest for practical applications.

Such numerical experiments related to real world machine learning regression data sets were done in [118, Section 4.2]: For this the data sets listed in Table 4.1 were considered

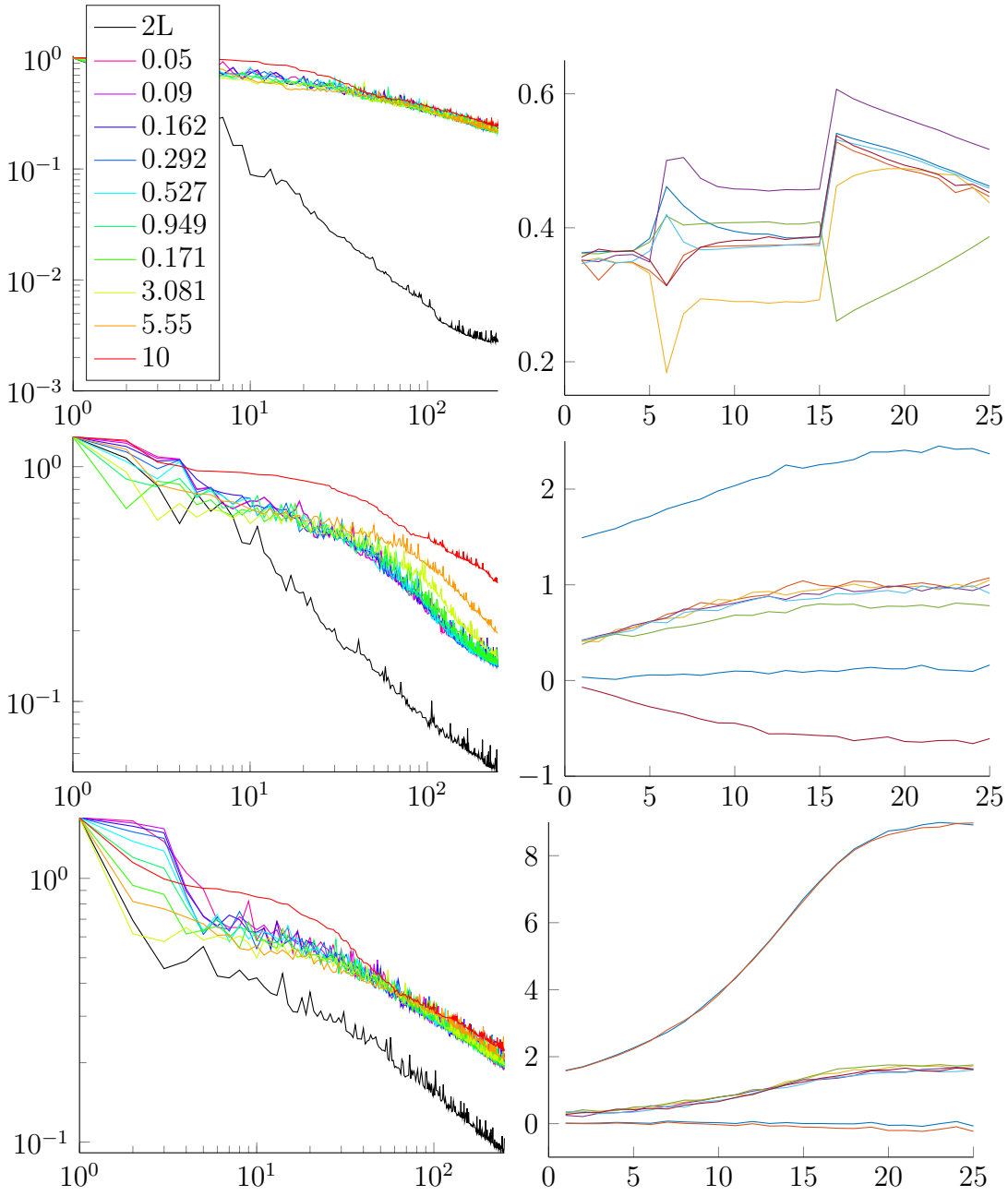

Figure 4.5: Top: 5D experiment, middle: 6D experiment, bottom: 7D experiment. Left: Visualization of the $\|\cdot\|_{L^\infty(\Omega)}$ error (y-axis) in the number of greedily selected points (x-axis) for both the standard kernel and the 2L-VKOGA. Right: Visualization of the change of the diagonal matrix entries as well as $A(2,1), A(1,2)$ of the matrix A_θ (y-axis) during the optimization epochs (x-axis). Figure taken from [118].

for the same numerical experiments as before (i.e. comparing approximation with 2L-VKOGA against standard shape parameter modified VKOGA). For this $n_{\text{epoch}} = 10$, $n_{\text{batch}} = 64$, $n_{\text{vkoga}} = 1000$ were used as well as a regularization of $\lambda = 10^{-3}$ for the kernel optimization and a regularization of $\lambda = 10^{-4}$ for the VKOGA (see [95]). The kernel is again the basic Matérn kernel. The results are depicted in Figure 4.6 and Figure 4.7, where one can see the test mean squared error for intermediate sizes of the model (up to an expansion size n_{vkoga}):

- For the datasets ct, sgemm, wecs, mlr_knn_rng, fried and kegg_undir_uci (see Figure 4.6) one can observe that the 2L-VKOGA model (black line) consistently outperforms the standard shape parameter modified kernels. In particular for several datasets, a given test accuracy can be achieved with a model expansion size which is an order of magnitude smaller.

- For the datasets diamonds, sarcos, stock, road_network, online_video as well as query_agg_count (see Figure 4.7) the 2L-VKOGA approach is not consistently better but shows mixed performances. Anyway the 2L-VKOGA approach is never significantly worse than any standard shape parameter modified kernel.

Recalling the cumulative energy of the singular values from Eq. (4.13) from Section 4.3.3, we observe a connection between the cumulative energy and the performance of the 2L-VKOGA algorithm: The faster the cumulative sum of Eq. (4.13) approaches the value one, the better the 2L-VKOGA algorithm performs. The plots within Figure 4.6 and Figure 4.7 where ordered according to the improvement achieved by the 2L-VKOGA. Like this it highlights the the connection to the cumulative energy as plotted in Figure 4.3.

4.3.5 Discussion and comments

We conclude this section with a discussion of the following points, which serve partly also as directions for future research:

1. (Dimensionality of A_θ): The dimensionality of the matrix A_θ is not intrinsically limited to $d \times d$. It is possible to consider $A_\theta \in \mathbb{R}^{b \times d}$, which is equal to an upsampling of features for $b > d$ or a downsampling for $b < d$.

   The first case of upsampling, i.e. $b > d$, is useless from the theoretical point of view: As $\operatorname{rank}(A_\theta) \leq \min(b, d) = d$, the effective dimension of the transformed dataset $X_N \cdot A_\theta^\top$ cannot increase. This means that the transformed dataset $X_N \cdot A_\theta^\top$ does not contain more information, i.e. no improved approximation using a two-layered kernel $A_\theta \in \mathbb{R}^{b \times d}$ is possible. An upsampling thus only makes sense, if a nonlinear mapping is considered – see the fourth point.

   The second case of downsampling, i.e. $b < d$ seems to make sense only in the setting described in Theorem 33 – i.e. if the considered function respective data (X_N, Y_N) is invariant in some directions. In this case, the matrix A_θ can reduce the dimensionality by removing the invariant directions. However, preliminary numerical experiments (not shown here) indicated that beyond those "invariant in some direction" cases, a downsampling (e.g. by removing the directions of the smallest singular values of an optimized matrix A_θ) is rather detrimental for the subsequent kernel model.

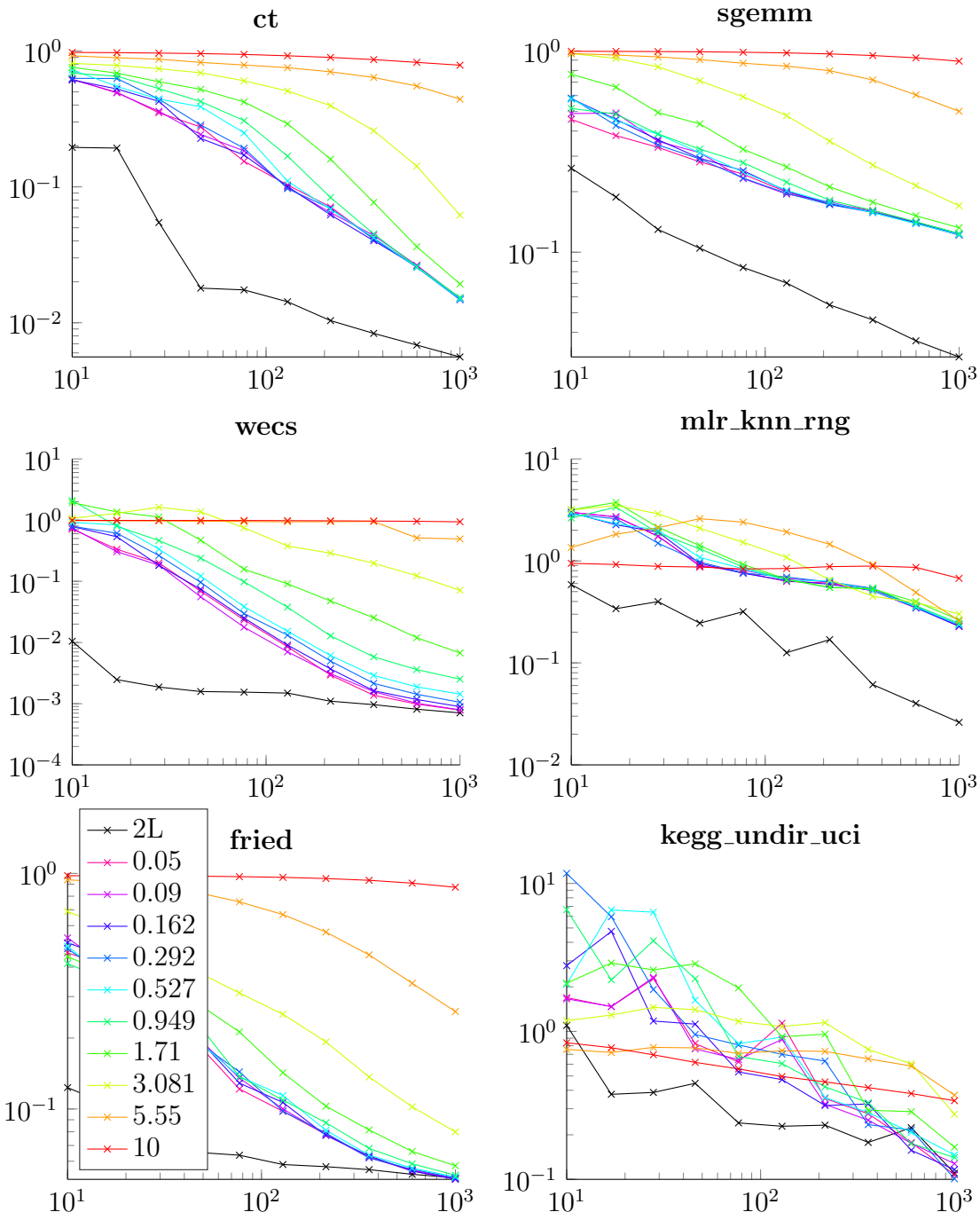

Figure 4.6: Visualization of the test MSE (y-axis) over the number of greedily selected centers (x-axis) for the first six datasets. The black line shows the 2L-VKOGA, while the colored lines show the use of standard kernels with different length scale parameters ε. Figure taken from [118].

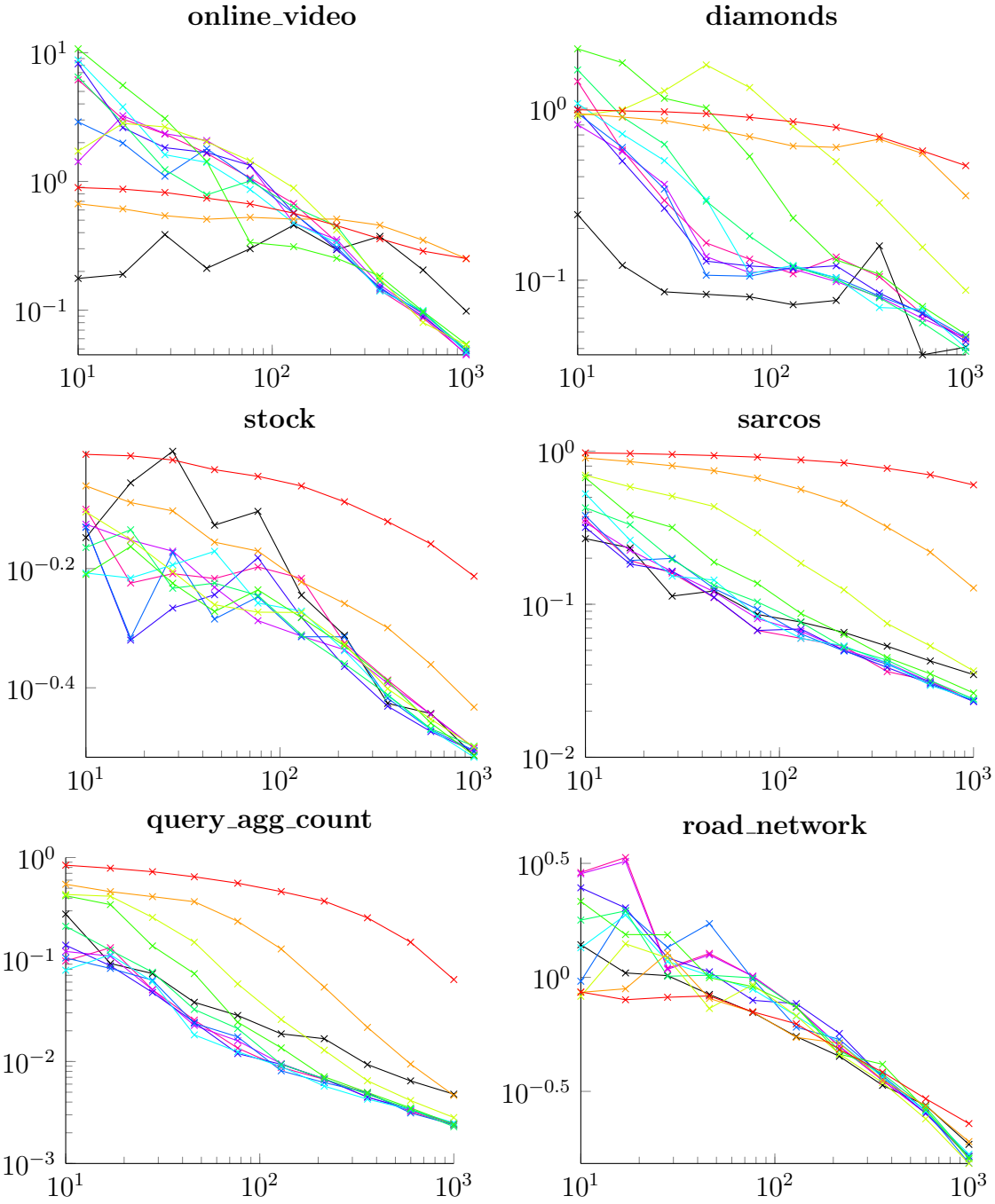

Figure 4.7: Visualization of the test MSE (y-axis) over the number of greedily selected centers (x-axis) for the last six datasets. The black line shows the 2L-VKOGA, while the colored lines show the use of standard kernels with different length scale parameters ε. Figure taken from [118].

2. (Least squares instead of interpolation): In the numerical experiments in Section 4.3.4, the optimized two-layered kernel is combined with greedy algorithms to derive accurate and sparse kernel models. In case the performance of the model is measured in the $\|\cdot\|_{L^2(\Omega)}$ error or a mean squared error, it makes sense to compute the final kernel model coefficients via a least squares fit instead of using the interpolation coefficients returned by running the greedy algorithm. This improves the accuracy even further. Such a least squares fit can be easily obtained even for huge datasets by leveraging existing software such as the FALKON algorithm [72,73], because the first layer matrix A_θ can be seen as a transformation of the input dataset.

3. (More than two-layered): In principle it is easily possible to use a deeper setup, i.e. more than two layers, and apply the same optimization procedure. However, we remark that it does not make sense to use several matrix-valued linear kernels after each other, because the corresponding linear transformations can be summarized into a single one. Assume L linear transformations with matrices $A_{\theta,1}, ..., A_{\theta,L}$, then they can be summarized into a single matrix and thus a single kernel mapping as $A_\theta := \prod_{i=1}^{L} A_{\theta,i}$.

4. (Nonlinear first layer kernels): Further possible extensions are to use nonlinear kernels in the first layer. However this turned out to be challenging. Both, too sophisticated kernels as well as simple polynomial kernels showed only poor or at most mixed performances.

4.4 Discussion, comments and outlook

In Section 4.2 and Section 4.3, two different kind of deep kernel models were introduced. The first approach – *SDKNs* – is a new machine learning model itself, which puts more focus on deriving a powerful kernel model resembling ideas from neural networks. The second approach – *two-layered kernels* – which can be seen as a generalization of hyperparameter optimized kernels, puts more focus on deriving a powerful kernel itself, which can subsequently be used in conjunction with e.g. standard greedy kernel methods. This is indeed one of the major differences between both approaches: The resulting deep kernel from Eq. (4.6) for the SDKN approach can not be used in conjunction with standard kernel methods such as greedy kernel algorithms – whereas the two-layered kernel of Eq. (4.9) can be used. The following points seem to be of importance for this:

1. (Optimization strategy): A randomized mini batch optimization was employed, that aimed at optimizing the kernel instead of directly calculating a suitable kernel model that fits the dataset. This second approach was followed in Section 4.2, where the loss from Eq. (4.7) aimed at directly optimizing the full deep kernel model.

2. (Structure of the kernel): The kernel k_θ of Eq. (4.9) is only a slight modification of a well understood RBF based kernel k. This is in contrast to Section 4.2, where the multiple layers including strong dimensionwise nonlinearities modified the inputs significantly before feeding them into the kernel of the final layer.
   Furthermore, also the kernel of the last layer is different: For the two-layered kernel approach, an RBF base kernel was used as the kernel of the last layer. For the

SDKN, the kernel of the last layer is a linear kernel – or when arguing that the linear kernel is only a linear combination of the previous features – a single-dimensional kernel. In any case, the expressivity of the last layer kernel of the SDKN is limited in comparison to a universal base kernel k for the two-layered approach.

Possible **future research** on the SDKN or the two-layered kernel approach was already discussed in Section 4.2.6 respective Section 4.3.5. Further future research on the combination and interplay of both approaches are the following directions:

- (Combination of both approaches): The SDKN models from Section 4.2 provide powerful kernel-based deep learning models while not giving a proper kernel as discussed in Section 4.2.3. The two-layered kernels of Section 4.3 provide proper kernels and if used in the 2L-VKOGA approach, also the resulting models are powerful – however due to the restricted first layer mapping in comparison to the SDKN approach, only little feature learning can be expected. In order to overcome these limitations it would be natural to extend the first layer mapping of the two-layered kernel approach, e.g. by using a SDKN-like setup here – while (partly) keeping the optimization idea of the two-layered kernels. Preliminary results indicated that such a straightforward combination does not work out. A possible reason for this could be the non-injectivity of the SDKNs, which were used as feature modifications. This could be alleviated by using invertible mappings, see e.g. [4] for invertible neural networks. Another more advanced idea could be a joint optimization, i.e. combining the loss of Eq. (4.7) (MSE loss) with the loss of the kernel optimization from the 2L approach Eq. (4.12).

- (Data-adapted kernel via NTK): In the introduction in Section 4.1 it was elaborated on the neural tangent kernel, which emerges from the training of infinite width neural networks. Recent research [50,131] focussed on different parametrization and scaling limits, which allow for feature learning also in the infinite width limit. Such a feature learning limit allows the corresponding NTK to learn features of the data. However, it remains unclear whether this can be used in a computational feasible way, as it is unclear whether or how such a kernel allows for a closed form expression or some economically computable function.

Chapter 5

Applications

One of the driving motivations of analyzing and improving kernel methods throughout this thesis was their potential for surrogate modeling purposes [45, 93, 127], which is the case for various scenarios in optimization or process control, where multi-query or real-time responses are required. The main goal of surrogate modeling is to replace some costly high-fidelity model by a surrogate model, which can be easily and quickly evaluated while still satisfying given accuracy demands. This corresponds to a trade-off between a minor loss in accuracy, but a huge benefit in terms of computational time.

The following Section 5.1 and Section 5.2 briefly showcase two important applications in the field of closure term prediction and the modeling of the human spine to demonstrate the usefulness of the novel approaches proposed in Section 4.2 and Section 4.3.

5.1 Closure term prediction

In order to showcase the applicability and flexibility of the introduced Structured Deep Kernel Networks (SDKNs) from Section 4.2, the application of SDKNs for the modeling of closure terms in turbulent flow simulations from [117] is presented here. The general idea hereby is, that in the field of computational fluid dynamics (CFD), often large amounts of high dimensional data occur. However, one is often only interested in a low dimensional quantity which can be extracted from this data, e.g. the drag of lift acting on a body within a flow. This is where machine learning tools come into play, as they are suitable to learn from large amounts of data. The reader is referred to [8–10] for more background information on the use of machine learning for turbulence modeling and to [102] for general background information on the mathematical description of fluid mechanics. Compressible fluid flows can be described by the Navier-Stokes equations (NSE), i.e.

$$U_t + R(F(U)) = 0 \tag{5.1}$$

in short and general form. Here $U = U(x, t)$ is the vector of conserved quantities that depends on both time t and location x. U_t denotes its time derivative, $F(U)$ the application of the nonlinear fluxes to U and $R(\cdot)$ denotes the divergence operator. Turbulence is generally characterized by the non-dimensional Reynolds number. Small Reynold numbers refer to less turbulent or even laminar flows, and an increasing Reynolds number leads to more turbulent flow, which exhibits chaotic behavior and an increasingly more

pronounced multiscale character from the small to large scales. Therefore, a solution of the full equation (e.g. via direct numerical simulation (DNS)) is unbearable for most practical applications, because most flows of interest are at least partly turbulent. To alleviate this challenge, large eddy simulation (LES) is employed: The governing equations from Eq. (5.1) are split into fine-scale and coarse-scale contributions applying a low-pass filter $\overline{(\cdot)}$. The exact choice of the low-pass filter here specifies the concrete method, and the effect of different filter functions on the full solution (DNS) can be seen in Figure 5.1. Eq. (5.1) is thus transformed into the coarse-scale governing equation

$$\overline{U}_t + \overline{R(F(U))} = 0.$$

However, as the flux F is nonlinear, the filtered flux term $\overline{R(F(U))}$ still depends on the solution U instead of on the coarse-solution $\overline{U}$ – this is the so-called *closure problem*. Using an appropriate numerical discretization $\tilde{R}(\overline{U})$ and adding $0 = \tilde{R}(\overline{U}) - \tilde{R}(\overline{U})$, we obtain

$$\overline{U}_t + \tilde{R}(\overline{U}) = \underbrace{\tilde{R}(\overline{U}) - \overline{R(F(U))}}_{\text{perfect LES closure}}. \tag{5.2}$$

The right-hand side of Eq. (5.2) is called *perfect LES closure*. However, the computation of the exact closure term depends on U, since due to the nonlinearity of F, the filter does in general not commute with the nonlinear fluxes, i.e. $\overline{R(F(U))} \neq R(F(\overline{U}))$. Therefore it is replaced by a model M which depends only on the coarse-scale solution $\overline{U}$.

$$\left(\tilde{R}(\overline{U}) - \overline{R(F(U))}\right) \approx M(\overline{U}). \tag{5.3}$$

These models used to be heuristic or physics- and mathematics-inspired, which however did not accurately enough capture the complexity of the closure terms, and no overall best model has been found yet. Nowadays, data-based machine learning models have been shown to be useful to recover these perfect closure terms.

For this task, also Structured Deep Kernel Networks (SDKNs) are suitable. In order to obtain a fair comparison with benchmark machine learning models from the literature, the same setup as in [8, 10] was used. In those publications, recurrent neural networks were used as machine learning models. In the following, we briefly recall the key settings and then elaborate on the results achieved by using SDKNs [117]:

The considered dataset stems from a high-fidelity DNS and is filtered by applying a projection filter, a top-hat filter as well as an Fourier filter, see Figure 5.1 for a visualization of their corresponding effects. The resulting dataset then consists of three-dimensional time series as inputs, namely the time-dependent coarse-scale filtered velocities $\overline{v}^{(1)}, \overline{v}^{(2)}, \overline{v}^{(3)}$ and the corresponding three-dimensional output time series given by the three closure terms $\overline{R(F(U))}_{1,2,3}$. However, only the final time step for the outputs is of interest here. The discretization of the (input) time series was given by the number of time samples (N_{seq}) and the time increment (Δt_{seq}) between successive steps. Three different choices were investigated, namely $(N_{\text{seq}}, \Delta t_{\text{seq}}) = (3, 10^{-3})$ (GRU1), $(N_{\text{seq}}, \Delta t_{\text{seq}}) = (10, 10^{-4})$ (GRU2), $(N_{\text{seq}}, \Delta t_{\text{seq}}) = (21, 10^{-4})$ (GRU3). As a preprocessing step, the (training) data was scaled according to a normal distribution, i.e. zero mean and unit variance.

As a baseline model, the neural network from [10] was considered, which is a recurrent neural network using a gated recurrent unit (GRU, [15]) module and the ReLU activation function. The input and output dimension are both 3 according to the data set description above, where the input uses 3, 10 or 21 time instances per sample depending on the choice of case GRU1, GRU2 or GRU3. Two hidden layers of dimension 32 and 64 were used before the GRU module, and two further hidden layers of dimension 48 and 24 were used after the GRU module. The optimization of the neural network was performed using a standard mini batch learning using the Adam optimizer with initial learning rate of 10^{-3} and a batch-size of 256 for 50 epochs. A step learning rate scheduler reduced the learning rate by a factor of 1/2 every 10 epochs. These learning settings for the NN were found to work well by empirical testing.

In order to have a fair comparison with the SDKN model, the same network structure was employed. For the optimization, only 25 epochs were used, such that the overall optimization time was approximately the same. The reason for this was the slightly more costly optimization steps due to the single dimensional kernel layers, that can be viewed as optimizable activation functions: Within those single dimensional kernel layers, the Gaussian kernel from Eq. (2.3) was used. Therefore, also the reduction of the learning rate was done every 5 instead of every 10 epochs, such that the final learning rate has been the same as for the NN case. As for the NN, the (same) GRU module was employed in the SDKN at the same position. We emphasize two points: First, the incorporation of the GRU module was easily possible due to the structured setup of the SDKN. Second, in contrast to the NN, no hyperparameter optimization was applied to the SDKN. Instead, the same training setup was adopted, which turned out to be highly suitable due to the flexibility of the SDKNs.

The performance of both the NN and the SDKN were evaluated via the mean squared error $\mathcal{L}_{\mathrm{MSE}}$ (MSE) and the Pearson's correlation coefficient $\mathcal{CC}$, that is defined as

$$\mathcal{CC}(a,b) = \frac{\mathrm{Cov}(a,b)}{\sqrt{\mathrm{Var}(a)} \cdot \sqrt{\mathrm{Var}(b)}}$$

using the covariance $\mathrm{Cov}(\cdot,\cdot)$ and the variance $\mathrm{Var}(\cdot)$. The mean squared error is given as

$$\mathcal{L}_{\mathrm{MSE}}(a,b) = \frac{1}{n} \cdot \sum_{i=1}^{n} \|a_i - b_i\|^2.$$

The results on the unseen test data are listed in Table 5.1: Throughout all the three test cases GRU1, GRU2 and GRU3 and the three applied filters (projection, top-hat and Fourier), the Pearson's correlation coefficient and the MSE-loss for the NN and the SDKN are very similar except for one case: For the combination of GRU2 and the projection filter, the NN performed significantly worse compared to the other cases (correlation coefficient of 0.8163 instead of usually 0.999, MSE-loss larger than 10^1 instead of smaller than $4{\cdot}10^{-1}$): We remark that this could be observed both in the original results within [10] as well as in the reproduced results in [117]. As the NN setup is large enough, it should be in theory capable of also learning a proper input-output mapping. However, for some unknown reason, the NN seems to get stuck during the optimization in a local minimum. Such a significant drop in performance is not observed for the SDKN. Instead, here the results

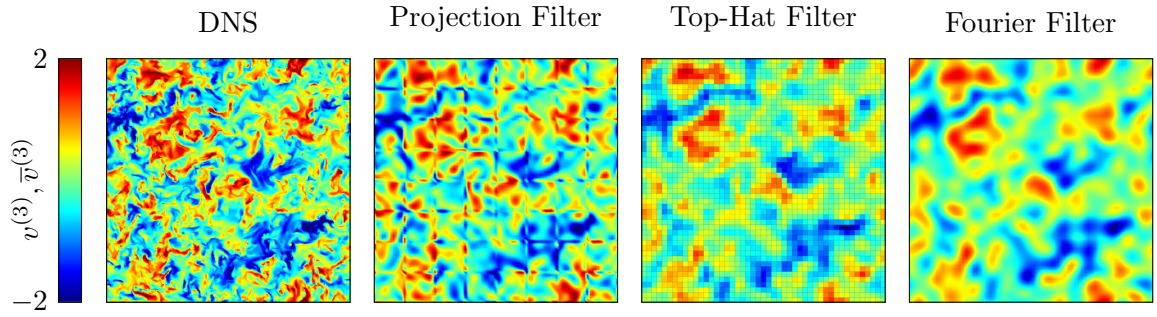

Figure 5.1: Two-dimensional slices of the three-dimensional z-velocity field $v^{(3)}$ for the high-fidelity DNS and for the corresponding filtered velocity fields $\overline{v}^{(3)}$ of the three investigated LES filters. Each slice of the filtered flow field contains 48^2 solution points. Figure taken from [117]

.

are similar to the results for the other filters. The convincing performance of the SDKNs supports again the increased flexibility of the SDKNs, which is likely due to the single dimensional kernel layers, that can be interpreted as optimizable activation functions. This can also be seen in the visualization within Figure 5.2, where two exemplary single-dimensional kernel layers are shown. On the left, a mapping using the Gaussian kernel (as in the experiments) is visualized, while on the right a mapping using the Wendland "$k = 0$" kernel (see Eq. (2.3)) is visualized.

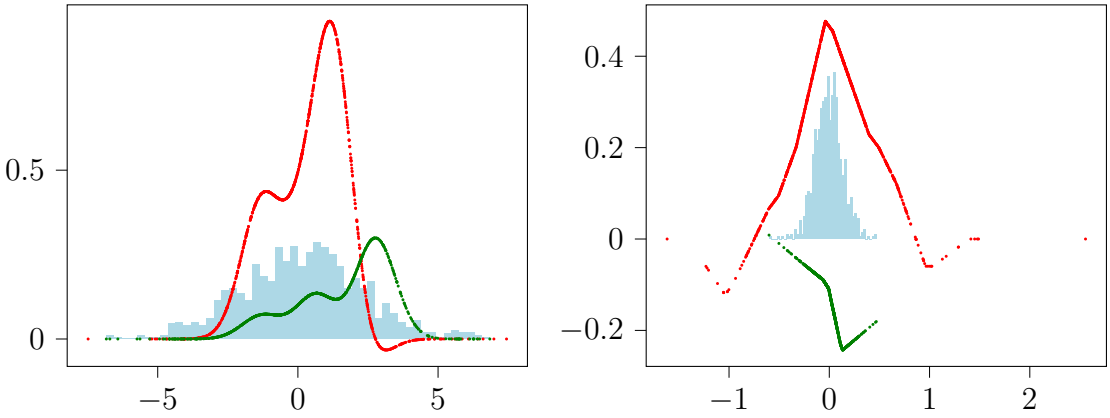

Figure 5.2: Visualization of two single-dimensional kernel mappings, that can be viewed as optimizable activation functions: The Gaussian kernel and the Wendland kernel (see Eq. (2.3)) are used on the left respective right. In red, the function before the optimization is visualized, while green visualizes the function after the optimization. The background histogram shows the distribution of the training data after the optimization. Figure taken from [117]

5.2 Spine modeling

While Section 5.1 showed the use of the SDKNs for surrogate modeling, here we highlight the use of greedy kernel models for this purpose in the context of spine modeling. We begin with a general introduction to the modeling of the human spine, and then continue

Table 5.1: Overview of the final MSE and Pearson's correlation coefficient on the unseen test data after optimization of the two machine learning models for the three test cases GRU1, GRU2, GRU3: Neural network (NN) and structured deep kernel network (SDKN).

MSE		GRU1	GRU2	GRU3
Projection	NN	3.235e-01	4.996e+01	3.233e-01
	SDKN	3.253e-01	3.261e-01	3.368e-01
Top-Hat	NN	3.155e-02	2.917e-02	2.888e-02
	SDKN	3.222e-02	2.989e-02	2.893e-02
Fourier	NN	1.179e-02	9.737e-03	9.452e-03
	SDKN	1.177e-02	1.007e-02	9.587e-03

Corr. coeff.		GRU1	GRU2	GRU3
Projection	NN	0.9989	0.8163	0.9989
	SDKN	0.9989	0.9989	0.9988
Top-Hat	NN	0.9992	0.9992	0.9992
	SDKN	0.9991	0.9992	0.9992
Fourier	NN	0.9992	0.9993	0.9993
	SDKN	0.9992	0.9993	0.9993

by showing the use of different (greedy) kernel methods:

- First in Section 5.2.1, the application of stabilized greedy methods is shown, which were originally introduced in [119]. These γ-stabilized greedy algorithms combine standard selection criteria as introduced in Section 2.5 (or more general β-greedy algorithms from Definition 18) with a stabilization parameter $\gamma \in (0, 1)$, that allows to derive more stable kernel models.

- Second in Section 5.2.2, the use of curl-free kernels for the modeling of the human spine is described, such that the resulting models ensure the physical requirement of energy conservation [47].

- Finally in Section 5.2.3, we comment on ongoing work, namely the combination of curl-free models explained before with the two-layered approach described in Section 4.3.

The human spine is an integral part of the human body and for its biomechanical analysis the vertebral bodies, ligaments, muscles and intervertebral discs (IVDs) are most important. In order to understand the biomechanical behavior of the human spine and derive accurate models for it (e.g. for the force-length characteristics of passive tissue compartments), its parts need to be modelled. The motivation for such a modeling is multiple fold, as those models can be used in a variety of applications such as injury prevention or medical research.

In this investigation, the focus was put on the intervertebral discs (IVDs), which connect two adjacent vertebral bodies. These allow for movements in six degrees of freedom (DOFs): Three translational and three rotational displacements, see Figure 5.3. For the accurate modeling of the IVDs including the nonlinear couplings of the different degrees of freedom, finite element models are used – which are however costly to compute

as well as to evaluate. In case the detailed and accurate modeling of a single IVD is not of utmost importance, but rather its biomechanical contribution to the overall model, surrogate models are a feasible option. For this, classical models like linear mappings (which are called stiffness-matrix or bushing element) or polynomial mappings were used [54, 55, 89]. However, especially linear approximations miss the nonlinear behavior and couplings of the stiffnesses of the IVDs. In order to improve the accuracy of such simple models on the one side, but still keep their speed and efficiency advantages on the other side, more sophisticated kernel models have been started to be used [127]. Especially when using target data-dependent greedy kernel methods, the derived models are even sparser and more efficient, in accordance with the analysis in Chapter 3.

5.2.1 Modeling via γ-stabilized greedy algorithms

The class of γ-stabilized algorithms [114, 119] modifies standard greedy selection criteria by restricting the set of possible inputs from which the next point is chosen from. This is done with the help of the power function $P_n(x)$: For a given selection criterion $\eta^{(n)}$, the next point is usually selected as

$$x_{n+1} := \arg\max_{x \in \Omega} \eta^{(n)}(x).$$

For the γ-stabilization, a value $\gamma \in [0, 1]$ is selected and the next point is selected from the restricted set $\Omega_\gamma^{(n)} := \{x \in \Omega \mid P_n(x) \geq \gamma \cdot \|P_n\|_{L^\infty(\Omega)}\}$ according to

$$x_{n+1} := \arg\max_{x \in \Omega_\gamma^{(n)}} \eta^{(n)}(x).$$

The motivation for this modification lies in the properties of those algorithms as analyzed in [119]. Due to using the restricted set $\Omega_\gamma^{(n)}$ for the selection of the next point, the new point x_{n+1} cannot be too close to previously chosen points X_n. This results in a stabilization effect, which means in practice that the computation does not terminate too early, see also Figure 5.4. More details on this class of stabilized algorithms, including the full analysis can be found in [119].

In [123], a particular data set resulting from the modeling of the IVDs was considered, where a symmetry in sagittal and frontal plane was used. This symmetry halved the number of independent components, such that the resulting dataset was three-dimensional in both input and output. It was presplit into 1238 training and validation points and 132 testing points and no preprocessing steps were required. For the computation of the numerical models, the linear Matérn kernel was used in conjunction with both the f-greedy and the f/P-greedy selection criterion. As base models, unstabilized kernel models were computed using a 5-fold cross-validation on the kernel shape parameter $\epsilon > 0$ (see Eq. (2.1)) and a subsequent 5-fold cross-validation on the regularization parameter $\lambda > 0$ (see Eq. (2.8)). For this, 20 logarithmically equally spaced values of ϵ in-between 10^{-2} and 10^{1} and 20 logarithmically equally spaced values of λ in-between 10^{-16} and 10^{3} were cross-validated. The stabilized models made use of the same shape parameter, and applied a 5-fold cross-validation instead to the stabilization parameter $\gamma \in [0, 1]$ and subsequently to the regularization parameter. For this, 11 equally spaced γ values in-between 0 and 1 were used as well as again 20 logarithmically equally spaced λ values in-between 10^{-16} and

10^3. In both cases, the algorithm is terminated when the residual drops below a threshold of 10^{-7} or the power function drops below 10^{-3} or if all training points are used.

The results of the best base model and best stabilized model for both the f-greedy and the f/P-greedy method are listed within Table 5.2. Both the selected hyperparameters as well as the resulting errors are presented, where the absolute and relative maximum as well as root-mean squared error (RMSE) were considered:

$$\begin{aligned} E_{\max,\mathrm{rel}} &:= \max_{i=1,..,|X|} \frac{\|s(x_i) - y_i\|_2}{\|y_i\|_2}, \\ E_{\mathrm{RMSE,rel}} &:= \left(\frac{1}{|X|} \cdot \sum_{i=1}^{|X|} \frac{\|s(x_i) - y_i\|_2^2}{\|y_i\|_2^2} \right)^{1/2}. \end{aligned} \tag{5.4}$$

Within Table 5.2 one can observe that the stabilized model always outperforms the base model: All the considered error measures are (partly by orders of magnitude) smaller for the stabilized models. From the listed hyperparameters, one can observe that this might be due to the increased expansion size of the kernel models: The calculation of the unstabilized kernel models terminates way earlier (142 compared to 690 and 63 compared to 358 centers) because the instability threshold due to τ_p is reached earlier. Thus, the stabilization allows defering the instability in favor of a larger expansion size, which allows obtaining more accurate models. This is also visualized within Figure 5.4: The left part of Figure 5.4 shows the number of selected centers within the five cross-validation runs. One can see that the number of selected centers increases with the stabilization parameter γ up to some saturation around $\gamma = 0.4$, where likely the stopping criteria due to τ_f is met. Thus, a (small) stabilization allows to obtaining a larger expansion size, while a full stabilization $\gamma = 1$ does not provide a larger expansion size than an intermediate stabilization (like $\gamma = 0.4$), however unnecessarily restricts the freedom to choose suitable centers due to $\Omega_\gamma^{(n)} \subset \Omega$. In the right part of Figure 5.4 the exemplary decay of the RMSE error in the number of selected centers is visualized: For $\gamma = 0$, i.e. the fully unstabilized algorithm, the expansion is stopped early. For $\gamma = 1$, i.e. the fully stabilized algorithm, the expansion is quite large – however, the corresponding error decay is rather slow. For the intermediate case of $\gamma = 0.2$ (which is also the optimal γ value as listed in Table 5.2), an intermediate expansion size with a proper decay rate of the error is obtained. Hence, this visualization again highlights the tradeoff between approximation rate and size of expansion.

	f-greedy			f/P-greedy	
	Hyperparameters	Results		Hyperparameters	Results
base	$\epsilon_{\text{base}} = 6.158 \cdot 10^{-2}$ $\gamma_{\text{base}} = 0$ $\lambda_{\text{base}} = 10^{-5}$ $n_{\text{base}} = 142$	$E_{\max} = 347.22$ $E_{\text{RMSE}} = 39.58$ $E_{\max,\text{rel}} = 6.95$ $E_{\text{RMSE,rel}} = 9.00 \cdot 10^{-1}$		$\epsilon_{\text{base}} = 4.281 \cdot 10^{-2}$ $\gamma_{\text{base}} = 0$ $\lambda_{\text{base}} = 10^{-2}$ $n_{\text{base}} = 63$	$E_{\max} = 4729.19$ $E_{\text{RMSE}} = 1104.12$ $E_{\max,\text{rel}} = 116.02$ $E_{\text{RMSE,rel}} = 14.83$
stabilized	$\epsilon_{\text{stab}} = \epsilon_{\text{base}}$ $\gamma_{\text{stab}} = 0.5$ $\lambda_{\text{stab}} = 10^{-5}$ $n_{\text{stab}} = 690$	$E_{\max} = 344.91$ $E_{\text{RMSE}} = 35.69$ $E_{\max,\text{rel}} = 1.79 \cdot 10^{-1}$ $E_{\text{RMSE,rel}} = 2.26 \cdot 10^{-2}$		$\epsilon_{\text{stab}} = \epsilon_{\text{base}}$ $\gamma_{\text{stab}} = 0.2$ $\lambda_{\text{stab}} = 10^{-15}$ $n_{\text{stab}} = 358$	$E_{\max} = 234.40$ $E_{\text{RMSE}} = 30.22$ $E_{\max,\text{rel}} = 6.77 \cdot 10^{-1}$ $E_{\text{RMSE,rel}} = 7.97 \cdot 10^{-2}$

Table 5.2: Overview of the selected hyperparameters and the error measures from Eq. (5.4) for the kernel models. Table taken from [46].

5.2.2 Modeling via curl-free kernels

While Section 5.2.1 mostly puts emphasis on modeling the input-output relation within the (training) data accurately, there can also be the requirement to model the underlying physics: In the case of models for the spine or in particular the IVDs, the underlying physics obeys in particular the conservation of the overall mechanical energy (kinetic energy and potential energy), as the motion of the spine model must not annihilate or produce any energy during its motions. This conservation of energy is equivalent to the existence of a potential $U : \mathbb{R}^d \to \mathbb{R}$ such that the forces $F_j, j = 1, ..., 6$ can be obtained as its gradients, i.e.

$$F_j = -\frac{\mathrm{d}U}{\mathrm{d}x^{(j)}}, \qquad j = 1, ..., d. \tag{5.5}$$

Following the ansatz of generalized interpolation as described in Section 2.4, an approximant s_U for U is built on the Riesz representer of those generalized interpolation conditions from Eq. (5.5). The Riesz representer of $(\delta_{x_i} \circ \frac{\mathrm{d}}{\mathrm{d}x^{(j)}})$ for $x_i \in \Omega$ is given by $w_{x_i}^{(j)} := D_2^j k(\cdot, x_i)$ – where the notation D_2^j refers to the derivative $\frac{\mathrm{d}}{\mathrm{d}x^{(j)}}$ applied to the second argument of the kernel. In order to satisfy Eq. (5.5) in the points x_i for $i = 1, ..., n$ we need

$$F_j(x_i) = -\frac{\mathrm{d}U}{\mathrm{d}x^{(j)}}(x_i) \qquad 1, .., N, \quad j = 1, ..., d,$$

and using the corresponding Riesz representer we obtain a model s_n for U as

$$s_n(\cdot) = \sum_{i=1}^{n} \sum_{j=1}^{d} \alpha_i^{(j)} D_2^j k(\cdot, x_i).$$

Taking derivatives, we obtain predictions for the forces F_j as

$$\begin{pmatrix} \frac{\mathrm{d}s_n}{\mathrm{d}x^{(1)}} \\ \vdots \\ \frac{\mathrm{d}s_n}{\mathrm{d}x^{(d)}} \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} \sum_{j=1}^{d} \alpha_i^{(j)} D_1^1 D_2^j k(\cdot, x_i) \\ \vdots \\ \sum_{i=1}^{n} \sum_{j=1}^{d} \alpha_i^{(j)} D_1^d D_2^j k(\cdot, x_i) \end{pmatrix}$$

$$= \sum_{i=1}^{n} \underbrace{\begin{pmatrix} D_1^1 D_2^1 k(\cdot, x_i) & \dots & D_1^1 D_2^d k(\cdot, x_i) \\ \vdots & \ddots & \vdots \\ D_1^d D_2^j k(\cdot, x_i) & \dots & D_1^d D_2^d k(\cdot, x_i) \end{pmatrix}}_{-k_{\text{curl free}}(\cdot, x_i)} \cdot \begin{pmatrix} \alpha_i^{(1)} \\ \vdots \\ \alpha_i^{(d)} \end{pmatrix}.$$

The matrix above can be interpreted as a matrix-valued curl-free kernel [27,28], such that the (vector-valued) prediction $(\frac{\mathrm{d}s_n}{\mathrm{d}x^{(1)}}, \dots, \frac{\mathrm{d}s_n}{\mathrm{d}x^{(d)}})^\top$ for the forces is directly a kernel model using this matrix-valued kernel. The curl-free kernel can be compactly written for any base kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ as

$$k_{\text{curl free}}(x, z) = -\nabla_1 \nabla_2^\top k(x, z) \in \mathbb{R}^{d \times d}, \tag{5.6}$$

where the index of the nabla operator refers to the argument to which it is applied.

Using this curl-free kernel, the theory and whole zoo of algorithms for greedy kernel algorithms can be applied, as most of the analysis for scalar-valued kernels carries over to matrix-valued kernels [128]. In particular, it was possible to use again stabilized greedy algorithms. For a detailed description of the data set, results and upcoming challenges we refer the interested reader to the publication [47].

5.2.3 Modeling via curl-free two-layered kernels

The kernel models introduced in Section 5.2.2 satisfy the requirement of energy conservation intrinsically due to the curl-free kernel from Eq. (5.6). On the other hand, for n data points in d dimensions, the resulting equation system will be of size $nd \times nd$ which can be challenging in terms of the stability of the greedy selection procedure. The stabilized algorithms used in the previous two sections can alleviate this partly. However, one is not restricted to using the greedy methods analyzed in Chapter 3, one can also make use of the deep kernel methods introduced in Chapter 4. In particular, it is possible to combine the curl-free kernel with the idea of a two-layered kernel and thus consider matrix-valued kernels of the form

$$k_{\text{curl free}}(x, z) = -\nabla_1 \nabla_2^\top k(A_\theta x, A_\theta z) \in \mathbb{R}^{d \times d}, \tag{5.7}$$

which is a generalization of the curl-free kernel from Eq. (4.9) by using $A_\theta = I_d$. Initial experiments using this approach were done in [84], which showed that tremendous speedups are possible in case the force field does not vary much in some directions. For the case of spine modeling with six DOFs, and thus a six dimensional force field, this can likely happen. The details of this approach are left for future work.

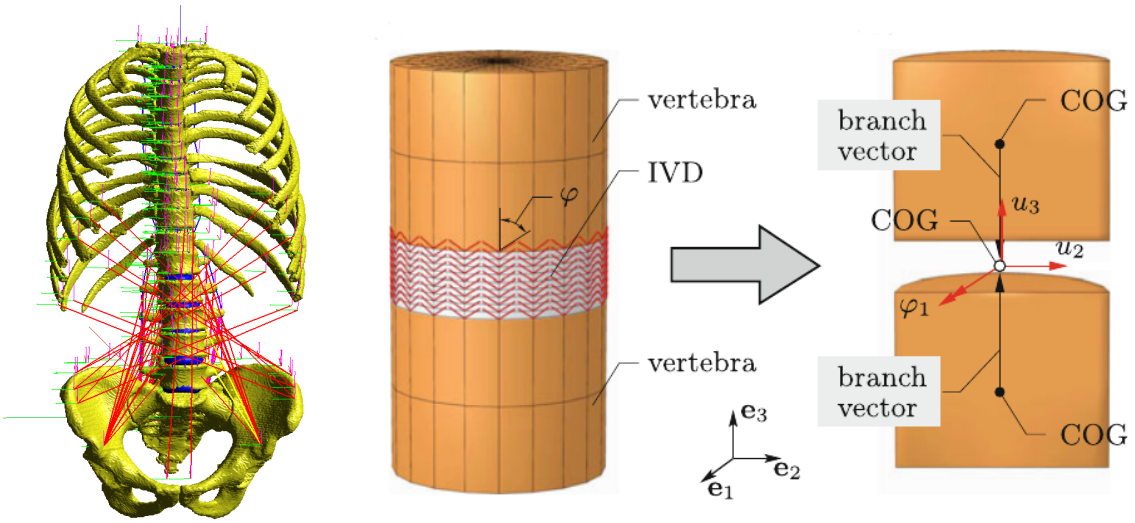

Figure 5.3: Visualization of the biomechanical model. On the left, the whole spine model is depicted, on the right the modeling scheme of an IVD reduced to a 3-d force/torque element is shown [55]. Figure taken from [46].

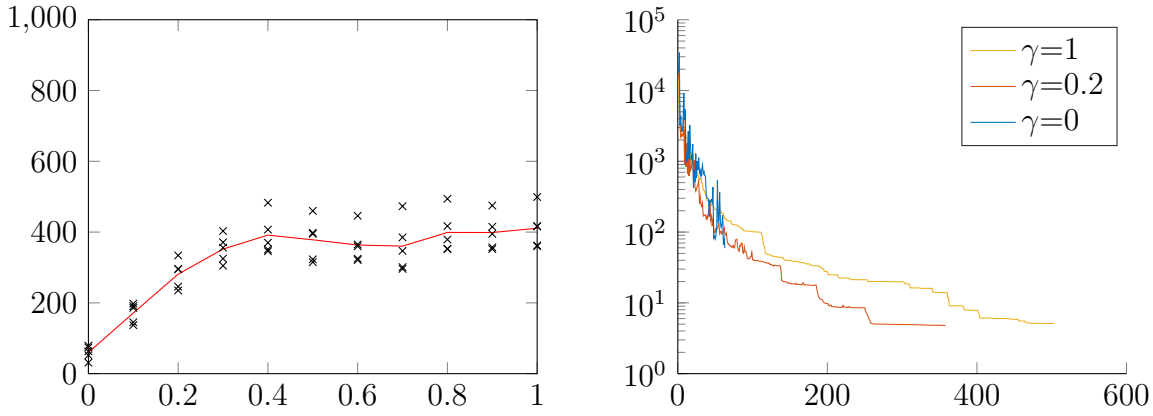

Figure 5.4: Visualization of the effect of the stability parameter for f/P-greedy on the number of chosen centers and the error decay: Left plot: Number of chosen centers (y-axis) over the stabilization parameter $\gamma \in [0, 1]$ (x-axis) for the 5 cross-validation runs (black crosses) and their mean value (red line). Right plot: RMSE error (y-axis) over the number of selected centers (x-axis) for three exemplary γ parameters. Figure taken from [46].

Chapter 6

Software

The experimental and computational work presented in this thesis was mainly done using python. The following list provides an overview on the software packages that have been extended and developed during this work:

1. **VKOGA extensions**: Code related to [122], see especially Section 3.5.1.
2. **ecVKOGA**: Code related to [47], see Section 5.2.
3. **PDE-VKOGA**: Code related to [123], see Section 3.6.
4. **2L-VKOGA**: Code related to [118], see Section 4.3.
5. **SDKN**: Code related to [120], see Section 4.2 and Section 5.1.

The software packages 1, 2, 3 and 4 are mainly based on the software package VKOGA [93][1]. While the software package 1 mostly adds new greedy algorithms (β-greedy and random greedy), the other packages indeed implement new techniques: The software package 2 implements greedy algorithms to derive energy conserving models (see Section 5.2), software package 3 implements algorithms to approximate the solutions of PDEs (see Section 3.6) and software package 4 extends the classical VKOGA by optimizing the two-layered kernel first (see Section 4.3).
Software package 5 provides an implementation of the SDKNs introduced in Section 4.2. The software packages 4 and 5 heavily make use of automatic differentiation, which is provided by using pytorch [83].

All the implementations can be found online at

`https://gitlab.mathematik.uni-stuttgart.de/pub/ians-anm`.

[1] See `https://github.com/GabrieleSantin/VKOGA`

Chapter 7

Conclusion

7.1 Summary

Machine learning techniques, which comprise the development of models and algorithms that learn with data, are ubiquitous in life nowadays. In recent years the subfield of deep learning was of particular interest due to its remarkable success in real world challenges. Nevertheless, more traditional kernel methods still provide reliable models for small to medium sized data sets, as they are based on a solid mathematical framework. This thesis presented and summarized advances in kernel-based methods from the analysis, algorithmic and application point of view. In particular, ideas and tools from deep learning were leveraged to introduce some kind of depth into the rather shallow kernel methods.

The first part of the thesis, i.e. Chapter 3, was more theoretical and showed improvements in the analysis and theory of standard kernel-based approximation: By generalizing an abstract framework for the analysis of greedy methods in Hilbert spaces, it was possible to derive strong utility statements as formulated in Section 3.3. Due to the convenient link, which was first established in [92] and then generalized in Section 3.4, it was possible to apply these results to kernel approximation and to show that $L^\infty(\Omega)$ convergence rates for kernel interpolation do not significantly depend on the shape of the domain, see Theorem 17 for a precise formulation. Section 3.5 proceeded by unifying greedy kernel algorithms from the literature into a joint framework of β-greedy algorithms and providing a joint analysis for them. In particular, it was proven for the first time that the target data-dependent f-greedy algorithm provides a better convergence rate than the target data-independent P-greedy algorithm, see Corollary 22. Section 3.7 discussed and commented on the overall findings of Chapter 3, and in particular, elaborated on possible future research directions related to kernel approximation using standard kernels.

The second part of the thesis, i.e. Chapter 4, introduced two different deep kernel models: First, so-called *structured deep kernel networks* (SDKNs) were introduced in Section 4.2, which combined simple classes of kernels in a structured way to put them into a multilayer setup. Theoretical statements showed their strong approximation properties, which were analyzed in three different asymptotic regimes: The number of centers, the width of the network and the depth of the network. Of particular importance was the limit of infinite depth, where kernel approximation properties from the "flat limit" of RBFs could be leveraged to show improved approximation constructions (see Theo-

rem 30) in comparison to corresponding constructions in the neural network's literature. The utility of these SDKNs was also highlighted in different numerical applications, see e.g. Section 5.1. Here, the structured setup of the SDKNs allowed for a combination with other common machine learning tools.
Second, two-layered kernels and the so-called *2L-VKOGA* (two-layered VKOGA) algorithm were introduced in Section 4.3, which combined a kernel optimization with a subsequent greedy algorithm to derive an efficient and sparse model. In comparison to the SDKN approach, this 2L-VKOGA did not only yield an efficient kernel model, but also a proper data-adapted kernel itself, which was due to a different optimization criterion: Instead of directly computing the final kernel model, first an optimization of the kernel itself was done with a subsequent computation of the final kernel model. The provided analysis on the optimized two-layered kernel gave insights into the underlying mechanics of the observed computational and experimental benefits of the methods.

The third part of the thesis, i.e. Chapter 5, gave an overview on two use cases of the introduced methods. The first application was about the challenging prediction of closure terms for turbulence flow simulation, where the used SDKNs were able to achieve the same accuracy as hyperparameter optimized neural networks and even outperformed them in some cases. The second application is related to the modeling of the human spine: Greedy kernel models were used as surrogate models for intervertebral discs, thus working as submodels in a larger model. Furthermore, an outlook on current work involving energy conserving models as well as the use of two-layered kernels for this task was provided.

7.2 Future work

Detailed comments on possible follow-up work were already provided in the respective discussion-and-comments sections, see in particular Section 3.7 for comments regarding (greedy) kernel approximation and Section 4.4 (as well as Section 4.2.6 and 4.3.5) for comments on deep kernel methods. Here, we briefly summarize those points from a high level point of view and elaborate on further points which concern the interplay of these topics:

For the kernel approximation research within Chapter 3, future research directions include improving the convergence rates and/or showing their optimality, extending the result to other norms and a closer investigation of analytic kernels, as most of the results considered finitely smooth kernels.

For the deep kernel research within Chapter 4, where structured deep kernels and two-layered kernels were introduced, a future research direction is how to combine these two approaches in a fruitful way: The SDKN are powerful expressive models, while the two-layered kernel optimization procedure results in a kernel that is useful in conjunction with standard kernel algorithms – e.g. greedy kernel methods.

Also, the interplay of the more theoretical work from Chapter 3 and the novel deep kernel models Chapter 4 can be investigated: The advantage of the two-layered kernel optimization (or even deeper kernels) in terms of the approximation error in the number of used data points can be further analyzed, in particular also for the non-asymptotic case which is of importance for practical applications. Moreover devising new greedy methods, which merge the benefits of kernel optimization and greedy selection in a joint

procedure seems to offer further improvements. Furthermore, the interplay of these novel results with other machine learning methods is of interest, e.g. the interplay with neural networks in terms of the neural tangent kernel: As the optimization of the two-layered kernels can be seen as a feature optimization, this might be related to a training of neural networks in a feature learning regime.

From the practical point of view – i.e. applying kernel methods to huge datasets – the literature already offers various approaches [1, 72, 73]. While greedy methods can be accelerated by introducing randomization [90], the combination of kernel optimization and scaling to large datasets is not yet investigated – however randomized methods also seem to be useful for this purpose.

Bibliography

[1] A. Abedsoltan, M. Belkin, and P. Pandit. Toward large kernel models. *arXiv preprint arXiv:2302.02605*, 2023.

[2] F. Aiolli and M. Donini. Learning anisotropic RBF kernels. In *International Conference on Artificial Neural Networks*, pages 515–522. Springer, 2014.

[3] P. D. Alfano, V. P. Pastore, L. Rosasco, and F. Odone. Fine-tuning or top-tuning? Transfer learning with pretrained features and fast kernel methods. *arXiv preprint arXiv:2209.07932*, 2022.

[4] L. Ardizzone, J. Kruse, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019.

[5] N. Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[6] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. *arXiv preprint arXiv:1910.01663*, 2019.

[7] S. Barthelmé and K. Usevich. Spectral properties of kernel matrices in the flat limit. *SIAM Journal on Matrix Analysis and Applications*, 42(1):17–57, 2021.

[8] A. Beck, D. Flad, and C.-D. Munz. Deep neural networks for data-driven LES closure models. *Journal of Computational Physics*, 398:108910, 2019.

[9] A. Beck and M. Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021.

[10] A. Beck and M. Kurz. A machine learning framework for LES closure terms. *Electronic Transactions on Numerical Analysis*, 56:117–137, 2022.

[11] A. Bietti and J. Mairal. On the inductive bias of neural tangent kernels. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] P. Binev, A. Cohen, W. Dahmen, R. DeVore, G. Petrova, and P. Wojtaszczyk. Convergence rates for greedy algorithms in reduced basis methods. *SIAM Journal on Mathematical Analysis*, 43(3):1457–1472, 2011.

[13] B. Bohn, C. Rieger, and M. Griebel. A representer theorem for deep kernel learning. *Journal of Machine Learning Research*, 20:1–32, 2019.

[14] C. Campi, F. Marchetti, and E. Perracchione. Learning via variably scaled kernels. *Advances in Computational Mathematics*, 47(4):1–23, 2021.

[15] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103, 2014.

[16] Y. Cho and L. K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

[17] P. G. Constantine. *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. Society for Industrial and Applied Mathematics, 2015.

[18] P. G. Constantine, E. Dow, and Q. Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[19] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

[20] A. Damianou and N. D. Lawrence. Deep Gaussian processes. In C. M. Carvalho and P. Ravikumar, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 207–215, Scottsdale, Arizona, USA, 2013.

[21] S. De Marchi, F. Marchetti, and E. Perracchione. Jumping with variably scaled discontinuous kernels (VSDKs). *BIT Numerical Mathematics*, 60(2):441–463, 2020.

[22] S. De Marchi, R. Schaback, and H. Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Advances in Computational Mathematics*, 23(3):317–330, 2005.

[23] R. DeVore, G. Petrova, and P. Wojtaszczyk. Greedy algorithms for reduced bases in Banach spaces. *Constructive Approximation*, 37(3):455–466, 2013.

[24] F. Dinuzzo. *Learning functions with kernel methods*. PhD thesis, University of Pisa Pisa, Italy, 2011.

[25] M. Dolbeault and A. Cohen. Optimal pointwise sampling for l^2 approximation. *Journal of Complexity*, 68:101602, 2022.

[26] F. Döppel, T. Wenzel, R. Herkert, B. Haasdonk, and M. Votsmeier. Goal-Oriented Two-Layered Kernel Models as Automated Surrogates for Surface Kinetics in Reactor Simulations. *Chemie Ingenieur Technik*, 2024.

[27] K. P. Drake, E. J. Fuselier, and G. B. Wright. A partition of unity method for divergence-free or curl-free radial basis function approximation. *SIAM Journal on Scientific Computing*, 43(3):A1950–A1974, 2021.

[28] K. P. Drake, E. J. Fuselier, and G. B. Wright. Implicit surface reconstruction with a curl-free radial basis function partition of unity method. *SIAM Journal on Scientific Computing*, 44(5):A3018–A3040, 2022.

[29] T. Driscoll and B. Fornberg. Interpolation in the limit of increasingly flat radial basis functions. *Computers & Mathematics with Applications*, 43(3):413–422, 2002.

[30] J. Duchon. Sur l'erreur d'interpolation des fonctions de plusieurs variables par les D^m-splines. *RAIRO. Analyse numérique*, 12(4):325–334, 1978.

[31] S. Dutta, M. W. Farthing, E. Perracchione, G. Savant, and M. Putti. A greedy non-intrusive reduced order model for shallow water equations. *Journal of Computational Physics*, 439:110378, 2021.

[32] R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940, Columbia University, New York, New York, USA, 2016.

[33] G. E. Fasshauer. *Meshfree Approximation Methods with MATLAB*, volume 6. World Scientific, 2007.

[34] G. E. Fasshauer and M. J. McCourt. *Kernel-based Approximation Methods using MATLAB*, volume 19. World Scientific Publishing Company, 2015.

[35] B. Fornberg and G. Wright. Stable computation of multiquadric interpolants for all values of the shape parameter. *Comput. Math. Appl.*, 48(5-6):853–867, 2004.

[36] J. H. Friedman. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67, 1991.

[37] T. Gao, S. Z. Kovalsky, and I. Daubechies. Gaussian process landmarking on manifolds. *SIAM Journal on Mathematics of Data Science*, 1(1):208–236, 2019.

[38] P. Gavrilenko, B. Haasdonk, O. Iliev, M. Ohlberger, F. Schindler, P. Toktaliev, T. Wenzel, and M. Youssef. A full order, reduced order and machine learning model pipeline for efficient prediction of reactive flows. In *Large-Scale Scientific Computing*, pages 378–386. Springer International Publishing, 2022.

[39] Q. L. Gia, F. Narcowich, J. Ward, and H. Wendland. Continuous and discrete least-squares approximation by radial basis functions on spheres. *Journal of Approximation Theory*, 143(1):124–133, 2006. Special Issue on Foundations of Computational Mathematics.

[40] A. Globerson and S. Roweis. Metric learning by collapsing classes. *Advances in Neural Information Processing Systems*, 18, 2005.

[41] J. Goldberger, G. E. Hinton, S. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. *Advances in Neural Information Processing Systems*, 17, 2004.

[42] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[43] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[44] B. Haasdonk, H. Kleikamp, M. Ohlberger, F. Schindler, and T. Wenzel. A New Certified Hierarchical and Adaptive RB-ML-ROM Surrogate Model for Parametrized PDEs. *SIAM Journal on Scientific Computing*, 45(3):A1039–A1065, 2023.

[45] B. Haasdonk and G. Santin. Greedy kernel approximation for sparse surrogate modeling. In W. Keiper, A. Milde, and S. Volkwein, editors, *Reduced-Order Modeling (ROM) for Simulation and Optimization: Powerful Algorithms as Key Enablers for Scientific Computing*, pages 21–45. Springer International Publishing, Cham, 2018.

[46] B. Haasdonk, T. Wenzel, G. Santin, and S. Schmitt. Biomechanical Surrogate Modelling Using Stabilized Vectorial Greedy Kernel Methods. In F. J. Vermolen and C. Vuik, editors, *Numerical Mathematics and Advanced Applications ENUMATH 2019*, pages 499–508, Cham, 2021. Springer International Publishing.

[47] M. Hammer, T. Wenzel, G. Santin, L. Meszaros-Beller, J. P. Little, B. Haasdonk, and S. Schmitt. A new method to design energy-conserving surrogate models for the coupled, nonlinear responses of intervertebral discs. *Biomechanics and Modeling in Mechanobiology*, pages 1–24, 2024.

[48] B. Hamzi and H. Owhadi. Learning dynamical systems from data: a simple cross-validation perspective, part I: parametric kernel flows. *Physica D: Nonlinear Phenomena*, 421:132817, 2021.

[49] B. Hanin. Universal function approximation by deep neural nets with bounded width and ReLU activations. *Mathematics*, 7(10), 2019.

[50] B. Hanin and M. Nica. Finite depth and width corrections to the neural tangent kernel. *arXiv preprint arXiv:1909.05989*, 2019.

[51] D. Holzmüller, V. Zaverkin, J. Kästner, and I. Steinwart. A framework and benchmark for deep batch active learning for regression. *Journal of Machine Learning Research*, 24(164):1–81, 2023.

[52] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[53] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[54] N. Karajan, D. Otto, S. Oladyshkin, and W. Ehlers. Application of the polynomial chaos expansion to approximate the homogenised response of the intervertebral disc. *Biomechanics and Modeling in Mechanobiology*, 13:1065–1080, 2014.

[55] N. Karajan, O. Röhrle, W. Ehlers, and S. Schmitt. Linking continuous and discrete intervertebral disc models through homogenisation. *Biomechanics and modeling in mechanobiology*, 12:453–466, 2013.

[56] G. S. Kimeldorf and G. Wahba. A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *The Annals of Mathematical Statistics*, 41(2):495–502, 1970.

[57] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[58] A. V. Knyazev and M. E. Argentati. Principal angles between subspaces in an a-based scalar product: Algorithms and perturbation estimates. *SIAM Journal on Scientific Computing*, 23(6):2008–2040, 2002.

[59] D. Krieg and M. Ullrich. Function Values Are Enough for L_2-Approximation. *Foundations of Computational Mathematics*, 21(4):1141–1151, 2021.

[60] D. Krieg and M. Ullrich. Function Values Are Enough for L_2-Approximation: Part II. *Journal of Complexity*, 66:101569, 2021.

[61] L. Le, J. Hao, Y. Xie, and J. Priestley. Deep kernel: Learning kernel function from data using deep neural network. In *2016 IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT)*, pages 1–7, 2016.

[62] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. *arXiv preprint arXiv:1711.00165*, 2017.

[63] J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020:124002, Dec 2020.

[64] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993.

[65] G. Lorentz, M. von Golitschek, and Y. Makovoz. *Constructive Approximation: Advanced Problems*. Grundlehren der mathematischen Wissenschaften. Springer, 1996.

[66] S. Ma and M. Belkin. Diving into the shallows: A computational perspective on large-scale shallow learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3781–3790, Red Hook, NY, USA, 2017. Curran Associates Inc.

[67] J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 1399–1407, 2016.

[68] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*, pages 2627–2635, 2014.

[69] F. Marchetti. The extension of Rippa's algorithm beyond LOOCV. *Applied Mathematics Letters*, 120:107262, 2021.

[70] F. Marchetti and E. Perracchione. Efficient Reduced Basis Algorithm (ERBA) for kernel-based approximation. *Journal of Scientific Computing*, 91(2):41, 2022.

[71] M. McCourt. Using Gaussian eigenfunctions to solve boundary value problems. *Advances in Applied Mathematics and Mechanics*, 5(4):569–594, 2013.

[72] G. Meanti, L. Carratino, E. De Vito, and L. Rosasco. Efficient hyperparameter tuning for large scale kernel ridge regression. In *International Conference on Artificial Intelligence and Statistics*, pages 6554–6572, 2022.

[73] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi. Kernel methods through the roof: handling billions of points efficiently. *Advances in Neural Information Processing Systems*, 33:14410–14422, 2020.

[74] F. N. Mojarrad, M. H. Veiga, J. S. Hesthaven, and P. Öffner. A new variable shape parameter strategy for rbf approximation using neural networks. *Computers & Mathematics with Applications*, 143:151–168, 2023.

[75] S. Müller. *Komplexität und Stabilität von kernbasierten Rekonstruktionsmethoden (Complexity and Stability of Kernel-based Reconstructions)*. PhD thesis, Fakultät für Mathematik und Informatik, Georg-August-Universität Göttingen, 2009.

[76] F. Narcowich, J. Ward, and H. Wendland. Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting. *Mathematics of Computation*, 74(250):743–763, 2005.

[77] F. Narcowich, J. Ward, and H. Wendland. Sobolev Error Estimates and a Bernstein Inequality for Scattered Data Interpolation via Radial Basis Functions. *Constructive Approximation*, 24(2):175–186, 2006.

[78] M. P. Otto and R. Izbicki. RFFNet: Scalable and interpretable kernel methods via Random Fourier Features. *arXiv preprint arXiv:2211.06410*, 2022.

[79] H. Owhadi and G. R. Yoo. Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47, 2019.

[80] A. Pandey, J. Schreurs, and J. A. Suykens. Robust generative restricted kernel machines using weighted conjugate feature duality. In *Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, July 19–23, 2020, Revised Selected Papers, Part I 6*, pages 613–624. Springer, 2020.

[81] A. Pandey, J. Schreurs, and J. A. Suykens. Generative restricted kernel machines: A framework for multi-view generation and disentangled feature learning. *Neural Networks*, 135:177–191, 2021.

[82] R. Parhi and R. D. Nowak. Banach space representer theorems for neural networks and ridge splines. *The Journal of Machine Learning Research*, 22(1):1960–1999, 2021.

[83] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[84] M. Peschken. Deep kernel models for energy conserving approximation. Bachelor thesis, University of Stuttgart, 2021.

[85] A. Pinkus. Approximation theory of the MLP model in neural networks. *Acta numerica*, 8(1):143–195, 1999.

[86] A. Pinkus. *n-Widths in Approximation Theory*. Ergebnisse der Mathematik und ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics. Springer Berlin Heidelberg, 2012.

[87] C. Rieger. *Sampling inequalities and applications*. PhD thesis, Universität Göttingen, 2008.

[88] M. Rossini. Interpolating functions with gradient discontinuities via variably scaled kernels. *Dolomites Research Notes on Approximation*, 11(2), 2018.

[89] T. Rupp, W. Ehlers, N. Karajan, M. Günther, and S. Schmitt. A forward dynamics simulation of human lumbar spine flexion predicting the load sharing of intervertebral discs, ligaments, and muscles. *Biomechanics and modeling in mechanobiology*, 14:1081–1105, 2015.

[90] G. Santin. Randomized selection of quasi-optimal sample points in Reproducing Kernel Hilbert Spaces. In preparation., 2023.

[91] G. Santin and et al. Continuous superconvergence for kernel-based interpolation. Personal communication, 2023.

[92] G. Santin and B. Haasdonk. Convergence rate of the data-independent P-greedy algorithm in kernel-based approximation. *Dolomites Research Notes on Approximation*, 10:68–78, 2017.

[93] G. Santin and B. Haasdonk. Kernel methods for surrogate modeling. In P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, editors, *Model Order Reduction*, volume 2. De Gruyter, 2021.

[94] G. Santin, T. Wenzel, and B. Haasdonk. On the optimality of target-data-dependent kernel greedy interpolation in Sobolev Reproducing Kernel Hilbert Spaces. *arXiv preprint arXiv:2307.09811*, 2023.

[95] G. Santin, D. Wittwar, and B. Haasdonk. Greedy regularized kernel interpolation. *arXiv preprint arXiv:1807.09575*, 2018.

[96] S. Scardapane, S. Van Vaerenbergh, S. Totaro, and A. Uncini. Kafnets: Kernel-based non-parametric activation functions for neural networks. *Neural Networks*, 110:19–32, 2019.

[97] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Advances in Computational Mathematics*, 3(3):251–264, 1995.

[98] R. Schaback. Native Hilbert spaces for radial basis functions I. In *New Developments in Approximation Theory: 2nd International Dortmund Meeting (IDoMAT)'98, Germany, February 23–27, 1998*, pages 255–282. Springer, 1999.

[99] R. Schaback. A greedy method for solving classes of PDE problems. *arXiv preprint arXiv:1903.11536*, 2019.

[100] R. Schaback and H. Wendland. Adaptive greedy techniques for approximate solution of large RBF systems. *Numerical Algorithms*, 24(3):239–254, 2000.

[101] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In D. Helmbold and B. Williamson, editors, *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[102] J. Serrin. Mathematical principles of classical fluid mechanics. *Fluid Dynamics I/Strömungsmechanik I*, pages 125–263, 1959.

[103] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudometrics. In *Proceedings of the twenty-first International Conference on Machine Learning*, page 94, 2004.

[104] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[105] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[106] J. A. Suykens. Deep restricted kernel machines using conjugate feature duality. *Neural computation*, 29(8):2123–2163, 2017.

[107] M. Telgarsky. Benefits of depth in neural networks. In V. Feldman, A. Rakhlin, and O. Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016.

[108] V. N. Temlyakov. Greedy approximation. *Acta Numerica*, 17:235–409, 2008.

[109] M. Unser. A representer theorem for deep neural networks. *Journal of Machine Learning Research*, 20(110):1–30, 2019.

[110] G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

[111] H. Wendland. Sobolev-type error estimates for interpolation by radial basis functions. *Surface fitting and multiresolution methods*, pages 337–344, 1997.

[112] H. Wendland. *Scattered Data Approximation*, volume 17 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2005.

[113] H. Wendland and C. Rieger. Approximate interpolation with applications to selecting smoothing parameters. *Numerische Mathematik*, 101(4):729–748, 2005.

[114] T. Wenzel. Analytic properties and numerical investigations of restricted greedy kernel methods. Master thesis, University of Stuttgart, 2019.

[115] T. Wenzel. Sharp inverse estimates for radial basis function interpolation: One-to-one correspondence between smoothness and approximation rates. *arXiv preprint arXiv:2306.14618*, 2023.

[116] T. Wenzel, B. Haasdonk, H. Kleikamp, M. Ohlberger, and F. Schindler. Application of Deep Kernel Models for Certified and Adaptive RB-ML-ROM Surrogate Modeling. *arXiv preprint arXiv:2302.14526*, 2023. Accepted for LSSC 2023 proceedings.

[117] T. Wenzel, M. Kurz, A. Beck, G. Santin, and B. Haasdonk. Structured Deep Kernel Networks for Data-Driven Closure Terms of Turbulent Flows. In *Large-Scale Scientific Computing*, pages 410–418, Cham, 2022. Springer International Publishing.

[118] T. Wenzel, F. Marchetti, and E. Perracchione. Data-driven kernel designs for optimized greedy schemes: A machine learning perspective. *SIAM Journal on Scientific Computing*, 46(1):C101–C126, 2024.

[119] T. Wenzel, G. Santin, and B. Haasdonk. A novel class of stabilized greedy kernel approximation algorithms: Convergence, stability and uniform point distribution. *Journal of Approximation Theory*, 262:105508, 2021.

[120] T. Wenzel, G. Santin, and B. Haasdonk. Universality and Optimality of Structured Deep Kernel Networks. *arXiv preprint arXiv:2105.07228*, 2021.

[121] T. Wenzel, G. Santin, and B. Haasdonk. Stability of convergence rates: Kernel interpolation on non-Lipschitz domains. *arXiv preprint arXiv:2203.12532*, 2022. Accepted for publication in IMA Journal of Numerical Analysis.

[122] T. Wenzel, G. Santin, and B. Haasdonk. Analysis of target data-dependent greedy kernel algorithms: Convergence rates for f-, f· P-and f/P-greedy. *Constructive Approximation*, 57(1):45–74, 2023.

[123] T. Wenzel, D. Winkle, G. Santin, and B. Haasdonk. Adaptive meshfree solution of linear partial differential equations with PDE-greedy kernel methods. *arXiv preprint arXiv:2207.13971*, 2022.

[124] C. Williams. Computing with infinite networks. *Advances in Neural Information Processing Systems*, 9, 1996.

[125] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep kernel learning. In A. Gretton and C. C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pages 370–378, 2016.

[126] D. Wirtz and B. Haasdonk. A vectorial kernel orthogonal greedy algorithm. *Dolomites Research Notes on Approximation*, 6:83–100, 2013.

[127] D. Wirtz, N. Karajan, and B. Haasdonk. Surrogate modeling of multiscale models using kernel methods. *International Journal for Numerical Methods in Engineering*, 101(1):1–28, 2015.

[128] D. Wittwar. *Approximation with matrix-valued kernels and highly effective error estimators for reduced basis approximations*. PhD thesis, University of Stuttgart, 2022.

[129] Z.-M. Wu and R. Schaback. Local error estimates for radial basis function interpolation of scattered data. *IMA journal of Numerical Analysis*, 13(1):13–27, 1993.

[130] H. Xue, Z.-F. Wu, and W.-X. Sun. Deep spectral kernel learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4019–4025. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[131] G. Yang and E. J. Hu. Tensor programs IV: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pages 11727–11737, 2021.

Appendices

Appendix A

Proofs related to SDKNs

The proofs in the following sections are copied more or less directly from [117] with notation adapted to this thesis and some minor modifications.

A.1 Utility statements

Proof of Proposition 23. We consider $b = 1$. The extension to $b > 1$ is obvious. Thus $A \in \mathbb{R}^{b\times d}$ is simply a row vector and the linear mapping $x \mapsto Ax$ simplifies to a dot product, i.e. $Ax = \langle A^\top, x\rangle_{\mathbb{R}^d}$.
Using the linear kernel, the kernel map s reads

$$s(x) = \sum_{i=1}^{M} \alpha_i k(x, z_i) = \sum_{i=1}^{M} \alpha_i \langle x, z_i\rangle_{\mathbb{R}^d} = \langle x, \sum_{i=1}^{M} \alpha_i z_i\rangle_{\mathbb{R}^d}$$

Thus we have $Ax = s(x)$ iff there are $\alpha_i \in \mathbb{R}$ such that $\sum_{i=1}^{M} \alpha_i z_i = A^\top$. □

Proof of Proposition 24. We consider $b = 1$. The extension to $b > 1$ is obvious. Thus $\beta_j \in \mathbb{R}$ for $j = 1, .., d$. Set $\beta = (\beta_1, .., \beta_d) \in \mathbb{R}^d$ and decompose $\beta = \beta^{\|} + \beta^{\perp}$ with $\beta^{\|} \in \operatorname{span} g(\Omega) \subset \mathbb{R}^d, \beta^{\perp} \perp \operatorname{span} g(\Omega)$.

1. The mapping

$$(g^{(1)}(x), .., g^{(d)}(x))^\top \mapsto \sum_{j=1}^{d} \beta_j^{\|} g^{(j)}(x) = \langle \beta^{\|}, g(x)\rangle_{\mathbb{R}^d}$$

   can be realized as a kernel mapping, if the centers $z_1, .., z_d \in \Omega$ are chosen such that $\operatorname{span}\{g(z_i), i = 1, .., d\} \supset \operatorname{span} g(\Omega)$: Like this, we have $\beta^{\|} \in \operatorname{span} g(\Omega) = \operatorname{span}\{g(z_1), .., g(z_d)\}$ and Proposition 23 yields the representation via the linear kernel.

2. For $\beta^{\perp}$ we have

$$\sum_{j=1}^{d} \beta_j^{\perp} g^{(j)}(x) = \langle \beta^{\perp}, g(x)\rangle_{\mathbb{R}^d} = 0$$

   as $g(x) \in \operatorname{span} g(\Omega), \beta^{\perp} \perp \operatorname{span} g(\Omega)$.

□

A.2 Unbounded number of centers

Proof of Theorem 26. Using the Kolmogorov-Arnold theorem [65], we can represent the given function f as

$$f(x^{(1)},..,x^{(d)}) = \sum_{q=0}^{2d} \Phi\left(\sum_{j=1}^{d} \lambda_j \phi_q(x^{(j)})\right)$$

with continuous functions $\Phi : \mathbb{R} \to \mathbb{R}$ and continuous strictly increasing bijective functions $\phi_q : [0,1] \to [0,1]$ for $q = 0,..,2d$, and scalar values $\lambda_j > 0, j = 1,...,d$ which satisfy $\sum_{j=1}^{d} \lambda_j \leq 1$. For our construction, we make use of the function decomposition depicted in Figure A.1.

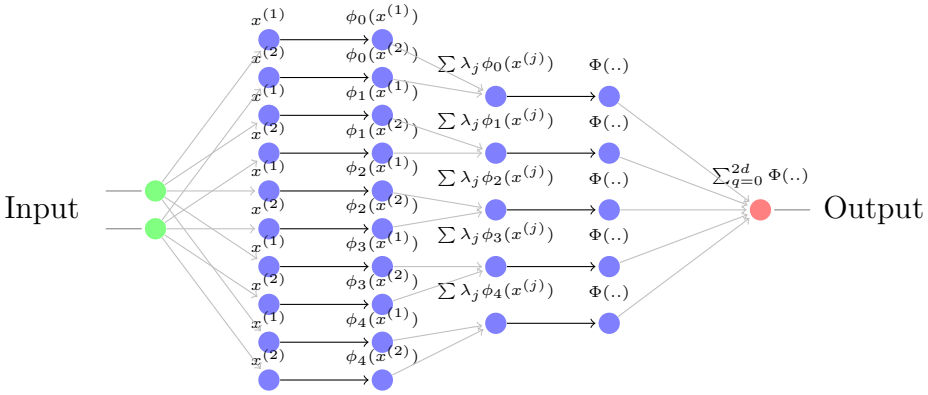

Figure A.1: Visualization of the unbounded number of centers case for $d = d_0 = 2, d_{2L+1} = 1$: Unbounded number of centers, but otherwise fixed setup. The first, third and fifth mapping use a linear kernel, the second and fourth one use a single dimensional kernels k_2 respective k_4. Within the linear kernel layers, not all connections are required. The first layer just duplicates the inputs, the second layer approximates the mappings $\phi_q(\cdot)$, the third layer builds the sum $\sum_{j=1}^{d} \lambda_j \phi_q(x^{(j)})$ and the fourth layer approximates the mappings Φ while the last layer builds the sum $\sum_{q=0}^{2d} \Phi(..)$. Figure taken from [117].

1. Approximation of $\phi_q : [0,1] \to [0,1]$: As the kernel k_2 is universal, it holds that the RKHS $\mathcal{H}_{k_2}([0,1]) \subset C([0,1])$ is a dense subspace in the continuous functions. By construction of the RKHS, it holds $\overline{\text{span}\{k_2(\cdot,u), u \in [0,1]\}} = \mathcal{H}_{k_2}([0,1])$. Therefore, since all $\phi_q, q = 0,..,2d$ are continuous, it holds

$$\forall \epsilon > 0 \; \exists N^{(q)} \in \mathbb{N}, \{u_i^{(q)}\}_{i=1}^{N^{(q)}} \subset [0,1], \{\alpha_i^{(q)}\}_{i=1}^{N^{(q)}} \subset \mathbb{R}$$
$$\text{such that } \left\| \phi_q(\cdot) - \sum_{i=1}^{N^{(q)}} \alpha_i^{(q)} k_2(\cdot, u_i^{(q)}) \right\|_{L^\infty([0,1])} < \epsilon.$$

We define the abbreviation $\sum_{i=1}^{N^{(q)}} \alpha_i^{(q)} k(\cdot, u_i^{(q)}) =: \tilde{\phi}_q(\cdot)$. As the argument before

holds for any $q = 0, .., 2d$, we have for all $x^{(j)} \in [0,1]$

$$\begin{aligned} \forall q = 0, .., 2d \qquad & \left\| \sum_{j=1}^{d} \lambda_j \phi_q(x^{(j)}) - \sum_{j=1}^{d} \lambda_j \tilde{\phi}_q(x^{(j)}) \right\|_\infty \\ & \leq \sum_{j=1}^{d} |\lambda_j| \left\| \phi_q(x^{(j)}) - \tilde{\phi}_q(x^{(j)}) \right\|_\infty \leq \epsilon \cdot \sum_{j=1}^{d} \lambda_j \leq \epsilon \end{aligned} \tag{A.1}$$

For any $q = 0, .., 2d$, $i = 1, .., N^{(q)}$ we define $u_i^{(q,0)} := u_i^{(q)} \cdot (1, .., 1)^\top \in \mathbb{R}^d$. Using this definition, we have $\mathbb{P}_j u_i^{(q,0)} = u_i^{(q)}$ for all $j = 1, .., d$, where $\mathbb{P}_j : [0,1]^d \to [0,1], x \mapsto x^{(j)}$ denotes the projection operator.
The points $\bigcup_{q=0}^{2d} \{u_i^{(q,0)}\}_{i=1}^{N^{(q)}} \subset [0,1]^d$ are used as centers.

2. Approximation of Φ on $I_q := (\sum_{j=1}^{d} \lambda_j \tilde{\phi}_q)([0,1]^d) \subset \mathbb{R}$ for $q = 0, .., 2d$: We employ the same reasoning as in the first step: The RKHS of k_4 is dense in the continuous functions, and as Φ is continuous on I_q it holds

$$\begin{aligned} \forall \epsilon > 0 \; \exists M^{(q)} \in \mathbb{N}, \{v_i^{(q)}\}_{i=1}^{M^{(q)}} \subset I_q, \{\beta_i^{(q)}\}_{i=1}^{M^{(q)}} \subset \mathbb{R} \\ \text{such that } \left\| \sum_{i=1}^{M^{(q)}} \beta_i^{(q)} k_4(\cdot, v_i^{(q)}) - \Phi(\cdot) \right\|_{L^\infty(I_q)} < \epsilon. \end{aligned} \tag{A.2}$$

We define the abbreviation $\sum_{i=1}^{M^{(q)}} \beta_i^{(q)} k_4(\cdot, v_i^{(q)}) =: \tilde{\Phi}_q(\cdot)$.
As I_q is the range of $\sum_{j=1}^{d} \lambda_j \tilde{\phi}_q(\cdot^{(j)})$ on $[0,1]^d$, there exists some $v_i^{(q,0)} \in [0,1]^d$ for every $v_i^{(q)} \in I_q$ such that $\sum_{j=1}^{d} \lambda_j \tilde{\phi}_q \left(\left(v_i^{(q,0)} \right)^{(j)} \right) = v_i^{(q)}$.

3. We use the abbreviation $\phi_{q,j}(\cdot) := \lambda_j \phi_q(\cdot)$ and $\tilde{\phi}_{q,j}(\cdot) := \lambda_j \tilde{\phi}_q(\cdot)$, $j = 1, .., d$. Combining the two previous steps:

$$\begin{aligned} & \Phi \left(\sum_{j=1}^{d} \phi_{q,j}(x^{(j)}) \right) - \tilde{\Phi}_q \left(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}) \right) \\ = & \underbrace{\Phi \left(\sum_{j=1}^{d} \phi_{q,j}(x^{(j)}) \right) - \Phi \left(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}) \right)}_{(*_1)} + \underbrace{\Phi \left(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}) \right) - \tilde{\Phi}_q \left(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}) \right)}_{(*_2)} \end{aligned}$$

($*_1$): The interval $(\sum_{j=1}^{d} \phi_{q,j}(x^{(j)}))([0,1]^d) \cup (\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}))([0,1]^d) \subset \mathbb{R}$ is a compact set. Therefore, Φ is uniformly continuous on this set. Thus the estimate from Eq. (A.1) of the first step allows to bound ($*_1$).

($*_2$): This difference is bounded by Eq. (A.2) from the second step of the proof. There, $\|\tilde{\Phi}_q(\cdot) - \Phi(\cdot)\|_{L^\infty(I_q)} < \epsilon$ was proven.

4. Finally steps one, two and three can be applied for all $q = 0, .., 2d$. For this, define $\tilde{f}(x) = \sum_{q=0}^{2d} \tilde{\Phi}_q(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}))$. Then it holds

$$\begin{aligned}\|f - \tilde{f}\|_{L^\infty([0,1]^d)} &= \left\| \sum_{j=0}^{2d} \left(\Phi \left(\sum_{j=1}^{d} \phi_{q,j}(x^{(j)}) \right) - \tilde{\Phi}_q \left(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}) \right) \right) \right\|_{L^\infty([0,1]^d)} \\ &\leq \sum_{q=0}^{2d} \left\| \Phi \left(\sum_{j=1}^{d} \phi_{q,j}(x^{(j)}) \right) - \tilde{\Phi}_{q,j} \left(\sum_{j=1}^{d} \tilde{\phi}_{q,j}(x^{(j)}) \right) \right\|_{L^\infty([0,1]^d)} .\end{aligned}$$

   The estimate from the third step finishes the estimation.

The set of centers required for the approximation is finally given by the union of the centers within the steps above, that is

$$\bigcup_{q=0}^{2d} \left(\{u_i^{(q,0)}\}_{i=1}^{N^{(q)}} \cup \{v_i^{(q,0)}\}_{i=1}^{M^{(q)}} \right).$$

The realization of the linear combinations is possible based on Proposition 24. □

A.3 Unbounded width

For the proof of Theorem 28, we need the utility statements 34 and Theorem 35 (see [85, Theorem 3.2]):

Lemma 34. *Consider a continuous function $h \in C([a,b]), a < b$ and two points $z_1 < z_2 \in [a,b]$. Then it is possible to construct a decomposition $h = h_1 + h_2$ such that $h_1 : \mathbb{R} \to \mathbb{R}$ and $h_2 : \mathbb{R} \to \mathbb{R}$ have the following symmetry property:*

$$h_1(z_1 + x) = h_1(z_1 - x),\ h_2(z_2 + x) = h_2(z_2 - x), \quad \forall x \geq 0.$$

Sketch of proof of Lemma 34. We give a constructive proof and refer to Figure A.2: Define $d := |z_2 - z_1| > 0$. Subdivide the interval $I := [a,b]$ into pieces of length up to $2d$, starting at z_1: $I_l := [z_1 + 2ld, z_1 + 2(l+1)d]$ and $J_l := [z_1 - 2(l+1)d, z_1 - 2ld]$ for $l = 0, 1, ...$ Observe that I_l, J_l are reflected wrt. z_1. Then it holds $I = \left(\bigcup_{l \geq 0} I_l \cup \bigcup_{l \geq 0} J_l \right) \cap I$.
We define the functions h_1 and h_2 in an iterative fashion starting on the interval I_0 and continuing with the intervals $J_0, I_1, J_1, I_2, ...$:

1. Start: For $x \in I_0$ set $h_1(x) := h(x)$ and $h_2(x) := 0$.

2. Iteratively:
   For $x \in J_l, l \geq 0$:

   - Define $h_1(x)$ via the symmetry condition $h_1(z_1 - x) = h_1(z_1 + x)$.
   - Define $h_2(x) := h(x) - h_1(x)$ to satisfy $h(x) = h_1(x) + h_2(x)$.

   For $x \in I_l, l \geq 1$:

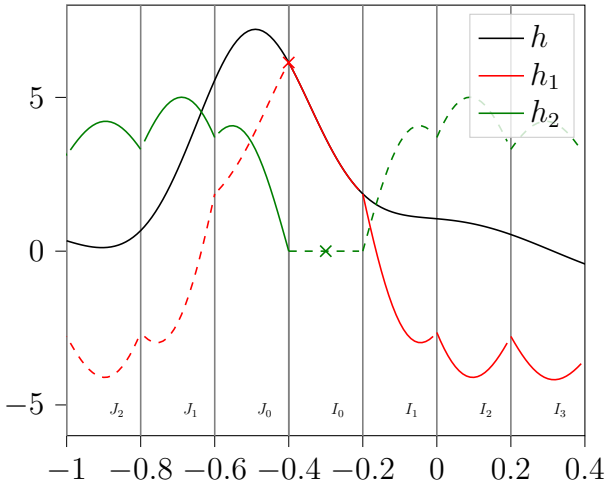

Figure A.2: Visualization of the decomposition of a function $h : \mathbb{R} \to \mathbb{R}$ (black) into two symmetric functions h_1 (red) and h_2 (green), such that it holds $h = h_1 + h_2$. Here we have $z_1 = -0.4$ and $z_2 = -0.3$ and thus $d = 0.1$. Figure adapted from [117].

- Define h_2 via the symmetry condition $h_2(z_2 + x) = h_2(z_2 - x)$.
- Define $h_1(x) := h(x) - h_2(x)$ to satisfy $h(x) = h_1(x) + h_2(x)$.

By construction it holds $h(x) = h_1(x) + h_2(x)$ for $x \in I$. The continuity of h_1, h_2 follows from the continuity of h via induction.

□

Theorem 35 (Special case of Vostrecov and Kreines, 1961)**.** *The space* $\operatorname{span}\{g(a^\top x) \mid g \in C(\mathbb{R}), a \in \mathcal{A}\}$ *for a given* $\mathcal{A} \subset \mathbb{R}^d$ *is dense in* $C(\mathbb{R}^d)$*, if* $\mathcal{A}$ *contains a (relatively) open subset of the unit sphere* $\mathbb{S}^{d-1} \equiv \{y \mid \|y\| = 1\}$.

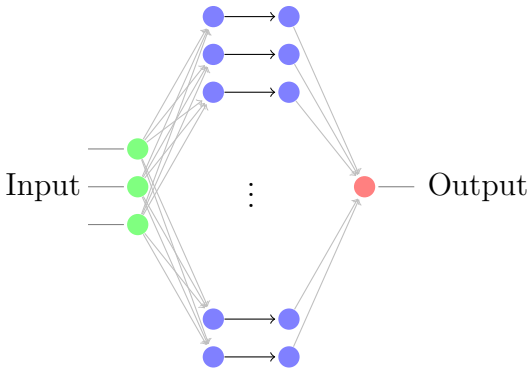

Figure A.3: Visualization of the unbounded width case for $d_0 = d = 3, d_1 = d_2, d_3 = 1$: Unbounded width, i.e. $d_1 = d_2 \to \infty$, but otherwise fixed setup. Figure taken from [117].

Proof of Theorem 28. The proof is split into several steps:

1. Based on a single center, we can approximate any given function that is symmetric with respect to that center on a finite interval:

Let φ be a radial basis function as specified in Assumption 27. As we only consider a single center for now, say z_1, we have

$$F_2^{(j)}(x) := (f_2 \circ f_1)(x)^{(j)} = \sum_{i=1}^{1} \alpha_{2,i}^{(i)} \varphi(|(A(x - z_i))^{(j)}|),\ 1 \leq j \leq d_1 = d_2$$

Now we choose the rows of the matrix A as positive multiples of each other, i.e. $\sigma_j \cdot a$, with $0 \neq a \in \mathbb{R}^{d_0}, \sigma_j \geq 0, i = 1, .., d_1 = d_2$. Then we have

$$\begin{aligned} F_2^{(j)}(x) &= \alpha_{2,1}^{(j)} \varphi(|\sigma_j a^\top (x - z_1)|),\ 1 \leq j \leq d_1 = d_2 \\ &= \alpha_{2,1}^{(j)} \varphi(|\sigma_j x_a|), \quad x_a := a^\top (x - z_1) \in \mathbb{R}, \end{aligned}$$

which is a scalar valued function in x_a. Now choose the last mapping f_3 such that a summation is realized, i.e. $A_3 = (1, 1, .., 1)^\top$, such that

$$s_1(x_a) := \sum_{j=1}^{d_2} \alpha_{2,1}^{(j)} \varphi(|\sigma_j x_a|). \tag{A.3}$$

This expression satisfies $s_1(-x_a) = s_1(x_a)$, thus only consider $x_a \geq 0$:

Due to Assumption 27 it holds that $\text{span}\{\varphi(\sigma x_a), \sigma \geq 0\}$ is dense in $C([0, 2\sqrt{d}\|a\|])$. Hence for sufficiently large d_2 and suitable coefficients $\alpha_{2,1}^{(j)}$ and $\sigma_j \geq 0$, the sum $s_1(x_a) = \sum_{j=1}^{d_2} \alpha_{2,1}^{(j)} \varphi(|\sigma_j x_a|), x_a \geq 0$ from Equation (A.3) approximates any function on $C([0, 2\sqrt{d}\|a\|])$ to arbitrary accuracy:

$$\forall_{\epsilon > 0, f \in C([0, 2\sqrt{d}\|a\|])} \quad \exists_{d_2 \in \mathbb{N}, \alpha_{2,1}^{(j)}, \sigma_j \in \mathbb{R}, j = 1, .., d_2}$$
$$\forall_{x_a \in [0, 2\sqrt{d}\|a\|]} \quad \left| f(x_a) - \underbrace{\sum_{j=1}^{d_2} \alpha_{2,1}^{(j)} \varphi(|\sigma_j x_a|)}_{s_1(x_a)} \right| < \epsilon.$$

For $x_a \in [-2\sqrt{d}\|a\|, 2\sqrt{d}\|a\|]$ the function $s_1(x_a)$ corresponds to a symmetric function in $C([-2\sqrt{d}\|a\|, 2\sqrt{d}\|a\|])$.

2. Leveraging Lemma 34, it is possible to approximate any given function that is univariate in direction $a \in \mathbb{R}^{d_0}$ on a finite interval:
First of all, according to Lemma 34, decompose $h \in C([-2\sqrt{d}\|a\|, 2\sqrt{d}\|a\|])$ into $h = h_1 + h_2$, whereby h_i satisfies $h_i(a^\top z_i + y) = h_i(a^\top z_i - y)$ for $y \geq 0, i = 1, 2$. Here we assume for now that both symmetry centers are different, i.e. $a^\top z_1 \neq a^\top z_2 \Leftrightarrow a^\top (z_1 - z_2) \neq 0 \Leftrightarrow a \not\perp z_1 - z_2$ is required. This requirement will be addressed in the next step. We have:

$$\begin{aligned} h(a^\top x) &= h_1(a^\top x) + h_2(a^\top x) \\ &= h_1(a^\top z_1 + a^\top (x - z_1)) + h_2(a^\top z_2 + a^\top (x - z_2)) \\ &=: \tilde{h}_1(a^\top (x - z_1)) + \tilde{h}_2(a^\top (x - z_2)) \end{aligned}$$

Both $\tilde{h}^{(1)}$ and $\tilde{h}^{(2)}$ are symmetric in their input with respect to zero. Therefore according to the first step, there exist s_1, s_2 such that

$$\forall y \in [-2\sqrt{d}\|a\|, 2\sqrt{d}\|a\|] \qquad |\tilde{h}_1(y) - s_1(y)| < \frac{\epsilon}{2}$$
$$\text{and } |\tilde{h}_2(y) - s_2(y)| < \frac{\epsilon}{2}.$$

Thus it is possible to estimate:

$$\begin{aligned} &|h(a^\top x) - s_1(a^\top(x - z_1)) - s_2(a^\top(x - z_2))| \\ \leq\ &|\tilde{h}_1(a^\top(x - z_1)) - s_1(a^\top(x - z_1))| + |\tilde{h}_2(a^\top(x - z_2)) - s_2(a^\top(x - z_2))| \\ \leq\ &\frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \end{aligned}$$

which holds for $x \in [0,1]^d$, as $|a^\top(x - z_i)| \leq 2\sqrt{d}\|a\|,\ i = 1, 2$.

3. Finally the approximation of arbitrary univariate functions from the second step is extended to the multivariate input case. For this, we leverage a theorem from Vostrecov and Kreines (see Theorem 35) and pick $\mathcal{A} := \{y \in \mathbb{S}^{d-1} \mid \|y - \frac{z_1 - z_2}{\|z_1 - z_2\|}\| < \frac{1}{2}\} \subset \mathbb{S}^{d-1}$, which is a relatively open subset. This choice of $\mathcal{A}$ yields

$$\begin{aligned} \frac{1}{4} > \left\| y - \frac{z_1 - z_2}{\|z_1 - z_2\|} \right\|^2 &= \|y\|^2 + \left\| \frac{z_1 - z_2}{\|z_1 - z_2\|} \right\|^2 - 2\left\langle y, \frac{z_1 - z_2}{\|z_1 - z_2\|} \right\rangle \\ \Leftrightarrow \quad \left\langle y, \frac{z_1 - z_2}{\|z_1 - z_2\|} \right\rangle &> 1 - \frac{1}{8} = \frac{7}{8} > 0, \end{aligned}$$

i.e. $a^\top z_1 \neq a^\top z_2$ for any $a \in \mathcal{A}$. Thus, the requirement $a^\top z_1 \neq a^\top z_2$ of the last step is satisfied.
Consider $f \in C([0,1]^d)$ arbitrary. For any $\epsilon > 0$, Theorem 35 gives the existence of $N \in \mathbb{N}, a_i \in \mathcal{A}, g_i \in C(\mathbb{R}), i = 1, .., N$ such that

$$\left| f(x) - \sum_{i=1}^{N} \alpha_i g_i(a_i^\top x) \right| < \frac{\epsilon}{2}.$$

Step 2 of the proof guarantees arbitrarily accurate approximation of those $\alpha_i g_i(a_i^\top x)$ for $x \in [0,1]^d$, i.e. $|a_i^\top x| \leq 2\sqrt{d}\|a_i\|$. This is possible, as $a_i^\top z_1 \neq a_i^\top z_2$, which is ensured by the choice of $\mathcal{A}$.

The realization of the linear combinations is possible based on Proposition 24. □

A.4 Unbounded depth

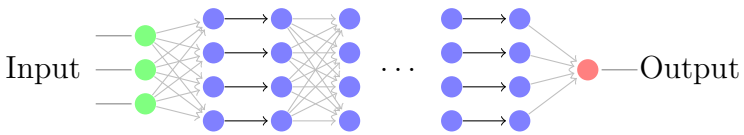

Figure A.4: Visualization of the unbounded depth case for $d_0 = 3, d_1 = .. = d_{L-1} = 4, d_L = 1$: Unbounded depth, i.e. $L \in \mathbb{N}$ arbitrary, but otherwise bounded width and number of centers. Figure taken from [117].

We recall a utility statement from the literature, which can be found in [29, Theorem 3.1].

Theorem 36. *Let $N \in \{2,3\}$ distinct data nodes $X_N = \{x_1,..,x_N\} \subset \mathbb{R}$ and corresponding target values $\{f_1,..,f_N\} \subset \mathbb{R}$ be given. Suppose the basis function*

$$\varphi(r) = \sum_{j=0}^{\infty} a_j r^{2j}$$

is strictly positive definite (i.e. the kernel $k(x,y) := \varphi(\epsilon\|x-y\|)$ is strictly positive definite). If

$$\begin{aligned} &a_0 \neq 0, a_1 \neq 0 && \text{in the case } N=2 \\ &a_1 \neq 0, 6a_0a_2 - a_1^2 \neq 0 && \text{in the case } N=3, \end{aligned} \tag{A.4}$$

then the kernel interpolant $s(x,\epsilon) = s(x)$ based on the nodes X_N satisfies for any $x \in \mathbb{R}$:

$$\lim_{\epsilon \to 0} s(x,\epsilon) = p_{N-1}(x). \tag{A.5}$$

Here, $p_{N-1}(x) \in \mathbb{P}_{N-1}$ is the interpolating polynomial for f on the nodes X_N.

Lemma 37 (Identity and squaring operation)**.** *Let $\Omega \subset \mathbb{R}_{\geq 0}$ be a compact interval. The functions $\psi_1 : \Omega \to \mathbb{R}, x \mapsto x$ and $\psi_2 : \Omega \to \mathbb{R}, x \mapsto x^2$ can be approximated by a SDKN satisfying the Assumptions 29 to arbitrary accuracy:*

$$\begin{aligned} &\operatorname{dist}(\psi_1(x) = x,\ \mathcal{F}_{1,1,3}(\Omega)) = 0 \\ &\operatorname{dist}(\psi_2(x) = x^2,\ \mathcal{F}_{1,1,3}(\Omega)) = 0 \end{aligned}$$

Proof. Since both the mappings ψ_1, ψ_2 are $\mathbb{R} \to \mathbb{R}$, we choose $d_0 = d_1 = d_2 = 1$. The linear mapping in the beginning acts only as a scaling $x \to \sigma \cdot x$, the linear mapping in the end is not required, i.e. its weight can be set to 1. Thus the output of the SDKN is given by

$$f_\sigma(x) = \sum_{i=1}^{3} \alpha_i k(\sigma x, \sigma z_i)$$

with $\alpha_i \in \mathbb{R}, \sigma > 0$ for a kernel k satisfying the requirements within Theorem 36. We choose α_i such that we have $f_\sigma(z_i) = \psi_j(z_i)$ for $i = 1,2$ $(j = 1)$ respective $i = 1,2,3$ $(j = 2)$. These are standard interpolation conditions that give the linear equation system (in case of $j = 3$)

$$\begin{pmatrix} k(\sigma z_1, \sigma z_1) & k(\sigma z_1, \sigma z_2) & k(\sigma z_1, \sigma z_3) \\ k(\sigma z_2, \sigma z_1) & k(\sigma z_2, \sigma z_2) & k(\sigma z_2, \sigma z_3) \\ k(\sigma z_3, \sigma z_1) & k(\sigma z_3, \sigma z_2) & k(\sigma z_3, \sigma z_3) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} \psi_2(z_1) \\ \psi_2(z_2) \\ \psi_2(z_3) \end{pmatrix}.$$

For $\sigma \to 0$ the flat limit is encountered: Leveraging Theorem 36 yields

$$\lim_{\sigma \to 0} f_\sigma(x) = p_2(x),$$

where $p_2(x)$ is the interpolating polynomial for the data $(z_1, \psi_j(z_1))$, $(z_2, \psi_j(z_2))$, $(z_3, \psi_j(z_3))$, which is given by $p_2(x) = x$ for $j = 1$ or $p_2(x) = x^2$ for $j = 2$. □

We remark that the mapped centers after the identity or squaring operation are still pairwise different due to the injectivity of ψ_1, ψ_2 on $\Omega \subset \mathbb{R}_{\geq 0}$, i.e. we have that $\psi_j(z_1)$, $\psi_j(z_2)$, $\psi_j(z_3)$ are pairwise distinct for both $j = 1, 2$.

Lemma 38 (Depth adjustment)**.** *Under the Assumptions 29, given an SDKN of depth L_1, it is possible to approximate this SDKN to arbitrary accuracy using another SDKN of depth $L_2 > L_1$:*

$$\forall w \in \mathbb{N}\ \forall f \in \mathcal{F}_{L_1,w,3}(\Omega)\ \operatorname{dist}(f, \mathcal{F}_{L_2,w,3}(\Omega)) = 0.$$

Proof. It suffices to prove the existence of a depth $L_2 - L_1$ SDKN that realizes an arbitrary accurate approximation $\tilde{I}_{d_0}$ of the identity mapping $I_{d_0} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_0}$: If this is the case, we have $\mathcal{F}_{L_1,w,3} \ni f = f \circ I_{d_0} \approx f \circ \tilde{I}_{d_0} \in \mathcal{F}_{L_2,w,3}$. The realization of such an approximation $\tilde{I}_{d_0}$ is possible due to Lemma 37 which states that $\psi_1(x) = x$ can be approximated to arbitrary accuracy. This can be employed in every input dimension $L_2 - L_1$ times. □

Lemma 39 (Product module)**.** *Let $\Omega \subset \mathbb{R}^2_{\geq 0}$ be a compact domain. The function $\psi : \Omega \to \mathbb{R}, (x, y) \mapsto xy$ can be approximated by an SDKN satisfying the Assumptions 29 to arbitrary accuracy:*

$$\operatorname{dist}(\psi(x, y) = xy,\ \mathcal{F}_{1,3,3}(\Omega)) = 0$$

Proof. In case the inputs x and y are linearly dependent (that means: the vectors $(x_i)_{i=1}^N \in \mathbb{R}^N, (y_i)_{i=1}^N \in \mathbb{R}^N$ (first and second dimension of the training data X_N) are linearly dependent), the output $xy = c \cdot x^2$ for some $c \in \mathbb{R}$ can be approximated by applying the squaring operation to one of its inputs with a proper scaling.
Thus assume linear independence in the input. The *product module* for this case is depicted in Figure A.5. We make use of

$$xy = \frac{1}{2\beta}\left[(x + \beta y)^2 - x^2 - \beta^2 y^2\right], \beta > 0 \tag{A.6}$$

The linear combination $(x, y) \to x + \beta y$ can be computed by the linear layer in the beginning, which is possible based on Proposition 23 due to the linear independence of the inputs. The linear combination of the squares is possible for the same reason. The parameter $\beta \neq 0$ is chosen such that the assumptions for the application of Lemma 37 are satisfied: It is required that $z_i^{(1)} + \beta z_i^{(2)}$, $i = 1, .., 3$ are pairwise distinct, which can be enforced based on $\beta \neq 0$ and the pairwise distinctness of $z_1^{(j)}, z_2^{(j)}, z_3^{(j)}$ for $j = 1, 2$. □

In the following we will waive to explicitly elaborate on the realization of the linear layers via Proposition 23. Even in case of linear dependency, all required linear layers can be realized as proven in Proposition 24.

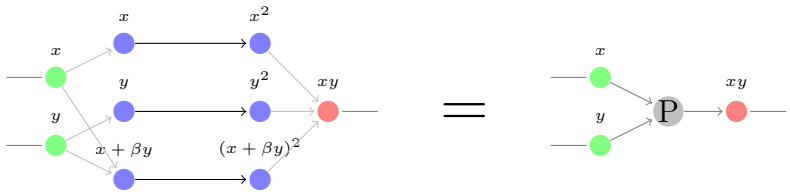

Figure A.5: Visualization of the product module (left) with corresponding abbreviation (right), which approximates the function $\psi(x, y) = xy$ based on Eq. (A.6) to arbitrary accuracy. The weights of the final linear layer of the left figure are each $\frac{1}{2\beta}$. Figure taken from [117].

Lemma 40 (Univariate monomial module)**.** *Let $\Omega \subset \mathbb{R}_{\geq 0}$ be a compact interval. The function $\psi : \Omega \to \mathbb{R}, x \mapsto x^n$ with $n \in \mathbb{N}$ can be approximated by a SDKN from $\mathcal{F}_{L,4,3}$ satisfying the Assumptions 29 to arbitrary accuracy:*

$$\operatorname{dist}(\psi(x) = x^n,\ \mathcal{F}_{L,4,3}(\Omega)) = 0$$

with depth $L = \max(\lceil \log_2(n) \rceil, 1)$.

Proof. The univariate monomial module is depicted in Figure A.6. The case $n = 1$ is simply the identity, whereas $n = 2$ corresponds to the squaring operation, which were treated in Lemma 37. Thus focus on $n \geq 3$: The approximation of $x^{2^{\lfloor \log_2(n) \rfloor}}$ can be realized easily by using the squaring operation from Lemma 37 $\lfloor \log_2(n) \rfloor$ times. The remaining factor $x^{n-2^{\lfloor \log_2(n) \rfloor}}$ can be multiplied (in case $n - 2^{\lfloor \log_2(n) \rfloor} \neq 0$) in the end by the multiplication operation: The factor $x^{n-2^{\lfloor \log_2(n) \rfloor}}$ was built in parallel and collected in the first dimension (i.e. in the top of Figure A.6) □

The assumption on the pairwise distinctness of the centers transfers through the whole monomial module, as the approximation of the squaring and the product module is exact in the respective centers, and the pairwise distinctness of z_1, z_2, z_3 implies the pairwise distinctness of z_1^n, z_2^n, z_3^n due to $\Omega \subset \mathbb{R}_{\geq 0}$.

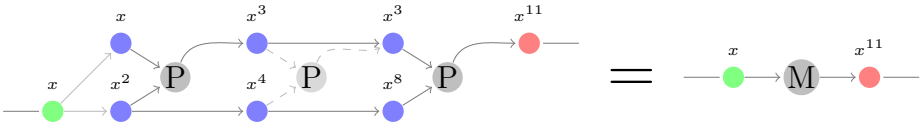

Figure A.6: Visualization of the monomial module (left) with corresponding abbreviation (right), which approximates the function $\psi(x) = x^n$ to arbitrary accuracy. Here, we have $n = 11$. The grey circles "P" refer to the product module. The dashed lines indicate the position of another potential product module, which is however not required as there is no x^4 contribution to build x^{11}. Figure taken from [117].

The following Lemma 41 shows that the approximation of polynomials in two inputs is possible, by combining the monomial and the product module.

Lemma 41 (Bivariate monomial module)**.** *Let $\Omega \subset \mathbb{R}^2_{\geq 0}$ be a compact domain. The function $\psi : \Omega \to \mathbb{R}, (x, y) \mapsto x^a y^b$ with $a, b \in \mathbb{N}$ can be approximated by a SDKN from $\mathcal{F}_{L,8,3}(\Omega)$ satisfying the Assumptions 29 to arbitrary accuracy:*

$$\operatorname{dist}(\psi(x, y) = x^a y^b, \mathcal{F}_{L,8,3}(\Omega)) = 0$$

with depth $L = \lceil \log_2(\max(a, b)) \rceil + 1$. Furthermore the following extension holds:

$$\operatorname{dist}(\tilde{\psi}(x, y, z) = x^a y^b + \beta z, \mathcal{F}_{L,9,3}(\Omega)) = 0, \ \beta \in \mathbb{R}.$$

Proof. The standard bivariate monomial module for the approximation of $\psi(x, y)$ is depicted in Figure A.7, top: Based on the two inputs x and y, the univariate monomial module is applied to each of them. In case of $a \neq b$, the depth of the univariate monomial modules is adjusted to $\max(a, b)$ using Lemma 38. This requires a depth of $L_1 = \max(\lceil \log_2(\max(a, b)) \rceil, 1)$. Subsequently, the product module from Lemma 39 is used to compute the final approximation of $x^a y^b$. Thus the total depth is given by $L = L_1 + 1$.
For the proof of the extension, i.e. the approximation of $\psi(x, y, z)$, the same construction can be extended with a final linear layer that adds the $+\beta z$ term. This is depicted in Figure A.7, bottom. □

It might happen that the centers collapse after the standard bivariate monomial module, i.e. that $\psi(z_1^{(1)}, z_1^{(2)})$, $\psi(z_2^{(1)}, z_2^{(2)})$, $\psi(z_3^{(1)}, z_3^{(2)})$ are no longer pairwise distinct. Therefore the extension to $\tilde{\psi}$ was introduced. As $\beta \in \mathbb{R}$ can be chosen arbitrarily, this alleviates the beforementioned center collapse.

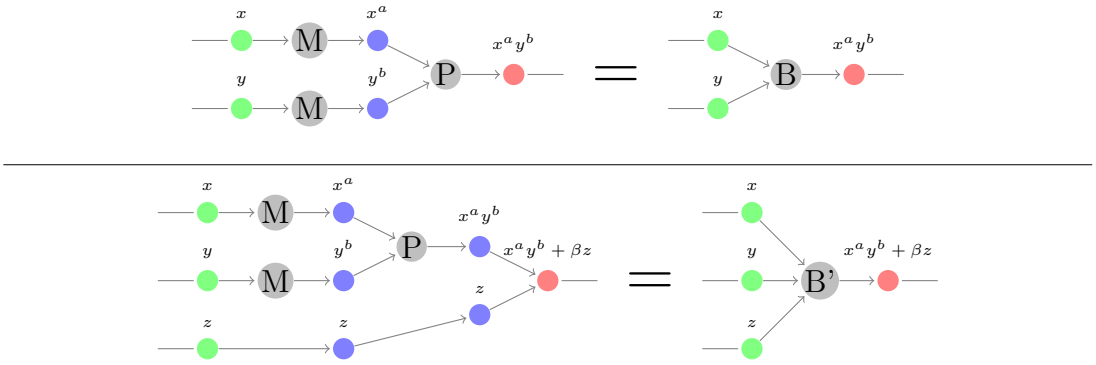

Figure A.7: Top: Visualization of the bivariate monomial module (left) with corresponding abbreviation (right), which approximates the function $\psi(x, y) = x^a y^b$ to arbitrary accuracy. The grey circles "M" refer to the univariate monomial module, whereas the grey circle "P" refers to the product module.
Bottom: Visualization for the extension, i.e. the approximation of $\tilde{\psi}$. Figure taken from [117].

Using these preliminary construction results, we can finally show how to add successively arbitrary polynomials to build a sum:

Lemma 42 (Addition module)**.** *Let $\Omega \subset \mathbb{R}^d_{\geq 0} \times \mathbb{R}$ be a compact domain. Under the Assumptions 29, a SDKN can approximate the mapping*

$$\psi : \Omega \to \mathbb{R}^{d+1}, \begin{pmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \\ S \end{pmatrix} \mapsto \begin{pmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \\ S' \end{pmatrix},$$

$$\text{with } S' := S + \alpha \cdot \prod_{j=1}^{d} (x^{(j)})^{n_j} + \beta x^{(d)}$$

to arbitrary accuracy with $\alpha, \beta \in \mathbb{R}, n_i \in \mathbb{N}$ arbitrary, i.e.

$$\operatorname{dist}(\psi(x), \mathcal{F}_{L,d+8,3}(\Omega)) = 0, \qquad L \leq d \cdot \lceil \log_2(\max_{i=1,..,d} n_i + 1) \rceil.$$

Remark 43. *The additional summand $\beta x^{(d)}$ is included for the same reason why we included the parameter β in the proof of Lemma 39: It might happen, that the used centers z_1, z_2, z_3 coincide after the mapping ψ, that is that $\psi^{(d+1)}(z_1), \psi^{(d+1)}(z_2), \psi^{(d+1)}(z_3)$ are no longer pairwise distinct. By including the β-summand, we can always find a $\beta \in \mathbb{R}$ such that the mapped centers $\psi(z_1)$, $\psi(z_2)$, $\psi(z_3)$ are pairwise distinct also in the dimension $d+1$.*

Proof of Lemma 42. The idea is to build the product $\prod_{j=1}^{d} (x^{(j)})^{n_j}$ in an iterative way: The first step starts with $(x^{(1)})^{n_1}$, then in the next steps the factors $(x^{(j)})^{n_j}, j = 2, .., d$ will be combined multiplicatively. Finally, the product is added to the sum. We assume $n_j > 0$ for $j = 1, .., d$, otherwise the inputs with $n_j = 0$ will simply be ignored.

1. The first step is realized with help of the univariate monomial module (and a linear layer contribution in case of $d > 1$) to build

$$r_1 := \begin{cases} (x^{(1)})^{n_1} & d = 1 \\ (x^{(1)})^{n_1} + \beta x^{(2)} & d > 1. \end{cases}$$

   This is depicted in Figure A.8 left. If $d = 1$, the next step is skipped. This step requires a depth of $L_1 = \max(\lceil \log_2(n_1) \rceil, 1)$, see Lemma 40.

2. In the k-th step, $k \in \{2, .., d-1\}$: We have $r_{k-1} = \prod_{j=1}^{k-1} (x^{(j)})^{n_j} + \beta x^{(k)}$. A bivariate monomial module multiplies the intermediate result r_{k-1} from the last step. Furthermore, a univariate monomial module builds $(x^{(k)})^{n_k+1}$. A subsequent linear layer combines these parts and the possible summand $\beta x^{(k+1)}$ to yield the intermediate result r_k:

$$\begin{aligned} r_k &= r_{k-1} \cdot (x^{(k)})^{n_k} - \beta (x^{(k)})^{n_k+1} + \beta x^{(k+1)} \\ &= \left(\prod_{j=1}^{k-1} (x^{(j)})^{n_j} + \beta x^{(k)} \right) \cdot (x^{(k)})^{n_k} - \beta (x^{(k)})^{n_k+1} + \beta x^{(k+1)} \\ &= \prod_{j=1}^{k} (x^{(j)})^{n_j} + \beta x^{(k+1)} \end{aligned}$$

The corresponding setup is depicted in Figure A.8, middle. The required depth L_k for this step is given by the depth of approximating $(x^{(k)})^{n_k+1}$, which is given by $L_k = \lceil \log_2(n_k + 1) \rceil$ due to Lemma 40 and $n_k > 0$.

3. The d-th step, where we have $r_{d-1} = \prod_{j=1}^{d-1}(x^{(j)})^{n_j} + \beta x^{(d)}$, simply multiplicatively combines the last contribution $(x^{(d)})^{n_d}$. Instead of an additional part like $\beta x^{(d+1)}$, we will add $\beta x^{(d)}$: A subsequent linear layer adds the product $r_d = \prod_{j=1}^{d}(x^{(j)})^{n_j}$ and the additional "$+\beta x^{(d)}$" to the sum S:

$$\begin{aligned} r_d &= r_{d-1} \cdot (x^{(d)})^{n_d} - \beta (x^{(d)})^{n_d+1} + \beta x^{(d)} \\ &= \left(\prod_{j=1}^{d-1} (x^{(j)})^{n_j} + \beta x^{(d)} \right) \cdot (x^{(d)})^{n_d} - \beta (x^{(d)})^{n_d+1} + \beta x^{(d)} \\ &= \prod_{j=1}^{d} (x^{(j)})^{n_j} + \beta x^{(d)} \end{aligned}$$

As explained in Remark 43, the "$+\beta x^{(d)}$ part is required to ensure that the three centers within the last dimension $d+1$ are pairwise distinct, i.e. $z_i^{(d+1)} + \prod_{i=1}^{d}(z_i^{(j)})^{n_j} + \beta z_i^{(d)}$ for $i = 1, 2, 3$ are pairwise distinct. This will be required in order to stack such sum-modules on top of each other later.
The required depth for this step can be infered from the bivariate monomial module (Lemma 41) and is thus given by $L_d = \max(\lceil \log_2(n_d) \rceil, 1)$.

Overall, the depth is limited by

$$L = \sum_{j=1}^{d} L_j \leq \sum_{j=1}^{d} \lceil \log_2(n_j + 1) \rceil \leq d \cdot \lceil \log_2(\max_{j=1,..,d} n_j + 1) \rceil.$$

□

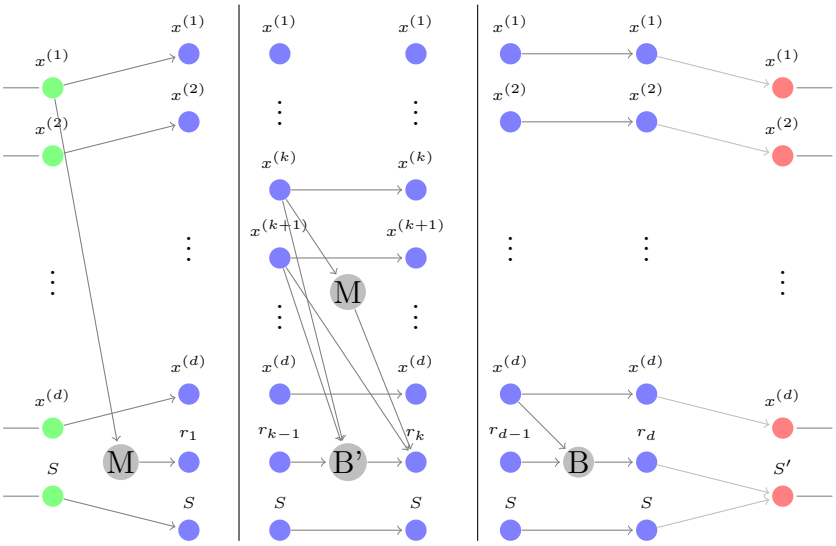

Figure A.8: Visualization of the construction in Lemma 42. Left: Adding a monomial in $x^{(1)}$. Middle: For $k \in \{2, .., d-1\}$, extend $r_{k-1} = \prod_{j=1}^{k-1}(x^{(j)})^{n_j} + \beta x^{(k)}$ to $r_k = \prod_{j=1}^{k}(x^{(j)})^{n_i} + \beta x^{(k+1)}$. Right: Extend $r_{d-1} = \prod_{=1}^{d-1}(x^{(j)})^{n_j} + \beta x^{(d)}$ to $r_d = \prod_{j=1}^{d}(x^{(j)})^{n_j}$ and add it to S. The grey circles "M", "B" and "B′" refer to the modules introduced earlier. Figure taken from [117].

Now it is time to state and prove the main result of this section, which is based on an iterative application of Lemma 42:

Proof of Theorem 30. By stacking the addition modules from Lemma 42 after each other, it is possible to approximate any polynomial $\sum_{l\in\mathbb{N}^d} \alpha_l x^l$ to arbitrary accuracy, where l is a multiindex. This means that $\bigcup_{L\in\mathbb{N}} \mathcal{F}_{L,d+8,3}$ is dense in the space of polynomials. As the space of polynomials is dense in the continuous functions (Stone-Weierstrass) and denseness is a transitive property, this implies that $\bigcup_{L\in\mathbb{N}} \mathcal{F}_{L,d+8,3}(\Omega)$ is dense in $C(\Omega)$. □

Appendix B

Proofs related to two-layered kernels

The proofs in the following are copied more or less directly from [118] and adapted to the notation of this thesis.

Proof of Theorem 32. We consider the two-layered kernel k_θ as a standard RBF kernel k applied to the transformed data $A_\theta X_n$ from the transformed domain $A_\theta\Omega$. As it holds $\operatorname{rank}(A_\theta) = d$ and Ω is a Lipschitz domain, we have $\dim(A_\theta\Omega) = \dim(\Omega) = d$. By standard Sobolev arguments (as $x \mapsto A_\theta x$ is just a linear full rank transformation) we have $H^\tau(\Omega) \asymp H^\tau(A_\theta\Omega)$, in particular $f \circ A_\theta^{-1} \in H^\tau(A_\theta\Omega)$. Furthermore, due to $\operatorname{rank}(A_\theta) = d$ all the singular values of A_θ are positive, i.e. it holds $s_{\min}(A_\theta)\|x\|_2 \leq \|A_\theta x\|_2 \leq s_{\max}(A_\theta)\|x\|_2$ or in short $\|\cdot\|_2 \asymp \|A_\theta \cdot\|_2$. Thus we obtain for the respective fill distances

$$\begin{aligned} h_{A_\theta\Omega, A_\theta X_n} &\equiv \sup_{\tilde{x}\in A_\theta\Omega} \left(\min_{\tilde{x}_j \in A_\theta X_n} \|\tilde{x} - \tilde{x}_j\|_2 \right) \\ &= \sup_{x\in\Omega} \left(\min_{x_j \in X_n} \|A_\theta x - A_\theta x_j\|_2 \right) \\ &\asymp \sup_{x\in\Omega} \left(\min_{x_j \in X_n} \|x - x_j\|_2 \right) = h_{\Omega, X_n}. \end{aligned}$$

Hence we can make use of the error bounds provided by Theorem 7 to derive the final statement. □

Proof of Theorem 33. Define $\mathcal{N} := \operatorname{Null}(\mathcal{A}_\theta)$ and consider the orthogonal projector $\Pi_{\mathcal{N}^\perp} : \mathbb{R}^d \longrightarrow \mathcal{N}^\perp$. The mapping $A_\theta^{\mathcal{N}^\perp} : \mathcal{N}^\perp \longrightarrow \mathrm{R}(A_\theta), x \mapsto A_\theta x$ is now full rank and thus invertible. Consider $x \in \Omega$ and decompose $x = x_\parallel + x_\perp$ with $x_\parallel \in \mathcal{N}, x_\perp \in \mathcal{N}^\perp$. Using the invariance assumption on f along $\mathcal{N}$, we have

$$\begin{aligned} |(f - s_{X_n})(x)| &= |f(x_\parallel + x_\perp) - \sum_{j=1}^n \alpha_j^{(n)} k(A_\theta(x_\parallel + x_\perp), A_\theta(x_{j,\parallel} + x_{j,\perp}))| \\ &= |(f(x_\perp) - \sum_{j=1}^n \alpha_j^{(n)} k(A_\theta x_\perp, A_\theta x_{j,\perp})| \\ &= |(f \circ (A_\theta^{\mathcal{N}^\perp})^{-1})(A_\theta^{\mathcal{N}^\perp} x_\perp) - \sum_{j=1}^n \alpha_j^{(n)} k(A_\theta x_\perp, A_\theta x_{j,\perp})|. \end{aligned} \tag{B.1}$$

As $\dim(A_\theta\Omega) \equiv d_{\text{eff}}$, the native space $\mathcal{H}_k(A_\theta\Omega)$ is now norm-equivalent to the Sobolev space $H^{\tau'}(A_\theta\Omega)$ of smaller smoothness $\tau' = \tau - \frac{d-d_{\text{eff}}}{2} < \tau$. Therefore we obtain $f \circ (A_\theta^{\mathcal{N}^\perp})^{-1} \in H^{\tau'}(A_\theta\Omega) \asymp \mathcal{H}_k(A_\theta\Omega)$.

Furthermore $\sum_{j=1}^n \alpha_j^{(n)} k(\cdot, A_\theta x_{j,\perp}) \in H^{\tau'}(A_\theta\Omega)$ and due to the kernel interpolation condition it holds

$$\sum_{j=1}^n \alpha_j^{(n)} k(y_i, A_\theta x_{j,\perp}) = (f \circ (A_\theta^{\mathcal{N}^\perp})^{-1})(y_i)$$

for all $y_i \in \{A_\theta x_i \mid x_i \in X_n\}$. Therefore we can leverage Theorem 7 to bound the error as

$$\left| (f \circ (A_\theta^{\mathcal{N}^\perp})^{-1})(y) - \sum_{j=1}^n \alpha_j^{(n)} k(y, A_\theta x_{j,\perp}) \right| < C h_{A_\theta\Omega, A_\theta X_n}^{\tau'-d_{\text{eff}}/2}, \qquad y \in \{A_\theta x \mid \boldsymbol{x} \in \Omega\}$$

$$\Leftrightarrow \quad |(f \circ (A_\theta^{\mathcal{N}^\perp})^{-1})(A_\theta^{\mathcal{N}^\perp} x_\perp) - \sum_{j=1}^n \alpha_j^{(n)} k(A_\theta x_\perp, A_\theta x_{j,\perp})| < C h_{A_\theta\Omega, A_\theta X_n}^{\tau'-d_{\text{eff}}/2}, \qquad x \in \Omega,$$

such that in conjunction with Eq. (B.1) we obtain

$$|(f - s_{X_n})(x)| \leq C h_{A_\theta\Omega, A_\theta X_n}^{\tau'-d_{\text{eff}}/2}, \qquad x \in \Omega.$$

Using finally $h_{A_\theta\Omega, A_\theta X_n} \asymp n^{-1/d_{\text{eff}}}$ which is possible due to $\dim(A_\theta\Omega) \equiv d_{\text{eff}}$ we obtain the desired statement:

$$-\frac{1}{d_{\text{eff}}} \cdot (\tau' - d_{\text{eff}}/2) = -\frac{\tau - \frac{d-d_{\text{eff}}}{2}}{d_{\text{eff}}} + \frac{1}{2} = -\frac{\tau}{d_{\text{eff}}} + \frac{d}{2d_{\text{eff}}}.$$

□

Declaration

I, *Tizian Wenzel*, declare herewith, that this work with the title

Deep and greedy kernel methods:
Algorithms, analysis and applications

is my own work. I declare that I am the sole author of this dissertation and that I have not reproduced, without acknowledgement, the work of another. I also declare that the electronic copy of this thesis coincides with the physical ones.

Place, Date	Tizian Wenzel