WWU

Fachbereich 10
Mathematik und
Informatik

Master Thesis

Kernel Interpolation: Theory and Applications

Celien Bosma

384092

First Supervisor: Prof. Dr. Mario Ohlberger
Second Supervisor: Prof. Dr. Xiaoyi Jiang

Westfälische Wilhems-Universität Münster

December 16, 2018

Contents

CHAPTER 1

Introduction

In this thesis, we are drawing a connection between numerical mathematics and learning theory by studying kernel interpolation. We will develop the requisite theory of kernels, pose a corresponding interpolation task, implement a particular kernel interpolation algorithm and use it for a specific application in learning theory.

Especially strictly positive definite kernels $K : \Omega \times \Omega \to \mathbb{R}$ play an important role in both fields, in numerical mathematics and in learning theory. In numerical mathematics, they are for example used in approximation and interpolation theory, as well as in meshless methods for solving partial differential equations. In learning theory they are often used as a tool to reduce computational cost, as for example in the context of feature maps. This could be the case when modeling a non-linear Support Vector Machine for example. Once a transformation Φ of features into a high-dimensional Hilbert space is needed to linearly separate classes in, the inner product

$$\langle \Phi(x), \Phi(y) \rangle =: K(x, y)$$

can be explicitly computed with the help of a kernel K. This technique is called the *kernel trick* and it allows to circumvent the computation of a potentially expensive computation of $\langle \Phi(\cdot), \Phi(\cdot) \rangle$ by cheap kernel evaluations. The authors of [12] give a wide overview of kernels and their applications in both numerical mathematics and learning theory and the authors of [13] study the matter more in-depth.

In chapter 2, we are going to establish the auxiliary theory behind kernels. First, we will introduce kernels and define strictly positive definiteness for them. In their simplest form, strictly positive definite kernels can be thought of as bell-shaped functions like Gaussians where $K(x, y)$ increases with similarity of x and y. Explicit forms of those kernels will be seen when we introduce Gaussian and Wendland kernels. Furthermore, we will prove that a strictly positive definite kernel $K : \Omega \times \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}^d$, induces an associated Hilbert space $\mathcal{N}_K(\Omega)$ of functions $f : \Omega \to \mathbb{R}$ in which every element $f \in \mathcal{N}_K(\Omega)$ can be reproduced by the kernel through

$$\langle f, K(\cdot, x) \rangle = f(x), \quad x \in \Omega.$$

This space is called the kernel's associated *Native Space*. We will see that for some kernels the associated Native Space is a Sobolev Space. The reproducing property in this Hilbert space will be convenient as it enables the numeric computation of the norm of functions $f \in \operatorname{span}\{K(\cdot, x) \mid x \in \Omega\}$ by computationally cheap kernel evaluations. Whereas the direct computation of the norm, e.g. for Sobolev Spaces, requires the expensive evaluation of integrals and derivatives.

In learning theory, one often faces classification or regression tasks in which a model needs to be determined that predicts the output y of given input x according to some empirical values

$(x_1, y_1), ...(x_N, y_N)$. This is often called *black box modeling* as there are no requirements on how the prediction is made, solely that it meets the passed empirical values. From a numerical point of view, this can be seen as finding a function s_f that approximates the (possibly unknown) model f according to

$$s_f(x_i) = y_i = f(x_i) \quad i = 1, ..., N. \tag{1.1}$$

Thus, this yields an interpolation problem on unstructured data $x_1, ..., x_N$. In chapter 3, we will search for a solution to this interpolation problem of the form

$$s_f = \sum_{i=1}^{N} \alpha_i K(\cdot, x_i)$$

for a fixed strictly positive definite kernel K and suitable coefficients $\alpha_1, ..., \alpha_N$. The linear system (1.1) determines the coefficients $\alpha_1, ..., \alpha_n$ and since the kernel K is strictly positive definite, a unique solution is guaranteed. However, the linear system (1.1) can be very large, non-sparse and ill-conditioned. That is why this approach is rarely implemented in practice. Instead, greedy algorithms are used that iteratively choose $n << N$ points $x_{j_1}, ..., x_{j_n}$ and compute the interpolants

$$f_n = \sum_{i=1}^{n} \alpha_i K(\cdot, x_{j_i}).$$

We will introduce one particular greedy algorithm in-depth, the so called *P-greedy algorithm*, and study it from a mathematical point of view when analyzing its convergence in chapter 4. We will relate its convergence to the Kolmogorov n-width and compare it to the best known convergence rate of non-greedy point selections. Then, in chapter 5, we will implement the P-greedy algorithm in python and try out our work under the best possible circumstances, a target model $f \in \text{span}\{K(\cdot, x) \mid x \in \Omega\}$.

Lastly, we will transfer our implementation to an application in learning theory in chapter 6. We will consider a standard classification task in pattern recognition, the recognition of handwritten digits. Here, we will use the free MNIST data base of 70,000 labeled 28×28 pixel black and white images of handwritten digits and try to identify the correct digit for each input image. This task is tackled really well by convolutional neural networks which first extract features by convolution and pooling for each input image and then classify by dense layers. The challenge for our P-greedy algorithm will be to approach this classification of feature vectors as a high-dimensional interpolation problem. For comparison, we have the ambitious accuracy of a convolutional neural network and hope our P-greedy algorithm can reach that accuracy.

CHAPTER 2

Kernels and their Native Spaces

In this chapter we will give an introduction to kernels and their associated Hilbert spaces of functions. This will yield the auxiliary tools to specify an algorithm for interpolating functions via kernels and to analyze the interpolation procedure elegantly in the underlying function Hilbert space. A thorough study of kernels and their associated reproducing kernel Hilbert spaces can be found in [16]. Our introduction will follow their concepts.

Let $\Omega \subset \mathbb{R}^d$ be a nonempty subset of $\mathbb{R}^d$, for any dimension $d \in \mathbb{N}$.

Definition 2.1. *We call a continuous function $K : \Omega \times \Omega \to \mathbb{R}$ a kernel on Ω.*

Definition 2.2. *A kernel is called (strictly) positive definite if it is symmetric and for all $N \in \mathbb{N}$ and point sets $X_N := \{x_1, ..., x_N\} \subset \Omega$ of pairwise distinct points in Ω the $N \times N$ symmetric kernel matrix*

$$A_{K,X_N} := (K(x_i, x_j))_{i,j=1,...,N}$$

is positive semi definite (resp. positive definite), i.e.

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \geq 0 \quad \textit{(resp. } >\textit{)}$$

for any coefficients $0 \neq (\alpha_1, ..., \alpha_N) \in \mathbb{R}^N$.

As we did not specify any restrictions on Ω, it could be finite. In this case, it would obviously not be possible to find pairwise distinct points for *any* $N \in \mathbb{N}$. In this case, we will mean all $N \in \mathbb{N}$ that allow the choice of N pairwise distinct points from Ω.

Analogously to the above definition, a general definition of (strictly) positive definiteness for complex valued functions $\Phi : \mathbb{R}^d \to \mathbb{C}$ can be formulated: Φ is (strictly) positive definite if for all $N \in \mathbb{N}$ and point sets $X_N := \{x_1, ..., x_N\} \in \mathbb{R}^d$ of pairwise distinct points the bilinear form

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \overline{\alpha_j} \Phi(x_i - x_j)$$

is non-negative (resp. positive) for all $0 \neq (\alpha_1, ..., \alpha_N) \in \mathbb{R}^N$. If Φ is real-valued, even and (strictly) positive definite in this sense, the symmetric kernel K_Φ defined by

$$K_\Phi(x, y) := \Phi(x - y)$$

is (strictly) positive definite in the sense of definition 2.2.

As we will see later, strictly positive definite kernels will play an important role in the interpolation process. Since the kernel matrix is positive definite for strictly positive definite kernels, it is invertible and ensures a unique solution to the interpolation problem. Moreover, a strictly positive definite kernel generates a certain kind of Hilbert space of functions $f : \Omega \to \mathbb{R}$, a so called *reproducing kernel Hilbert space*. Kernel interpolation exploits the structure of those function spaces and uses them to elegantly analyze the interpolation. We will introduce those spaces in the following sections.

2.1. Gaussian Kernels and Wendland Kernels

In this section we will introduce two classes of kernels, Gaussian kernels as well as Wendland kernels. In both cases, the kernels are generated by radial functions, i.e. they are of the form

$$K_\phi(x, y) = \phi(\|x - y\|_2), \quad x, y \in \mathbb{R}^d$$

for a function $\phi : [0, \infty) \to \mathbb{R}$. We will need some Fourier theory for this introduction. Therefore, we cited the most important definitions from chapter 5 in [16] in the appendix.

First, we will motivate Gaussian kernels. Our goal is to construct a strictly positive definite kernel for our interpolation process. For that we will need a famous result on positive definite functions $\Phi : \mathbb{R}^d \to \mathbb{C}$, which was proven in 1932-1933 by Bochner.

Theorem 2.3 (Bochner's characterization)**.** *A continuous function $\Phi : \mathbb{R}^d \to \mathbb{C}$ is positive definite if and only if it is the Fourier transform of a finite non-negative Borel measure μ on $\mathbb{R}^d$, i.e.*

$$\Phi(x) = \widehat{\mu}(x) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} e^{-ix^T\omega} d\mu(\omega) \quad x \in \mathbb{R}^d. \tag{2.1}$$

The idea behind this characterization is the following. Consider $\Phi \in \mathcal{C}(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$ and suppose it has an integrable Fourier transform $\widehat{\Phi} \in L_1(\mathbb{R}^d)$. Then, Φ can be recovered from its Fourier transform via

$$\Phi(x) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \widehat{\Phi}(\omega) e^{ix^T\omega} d\omega.$$

Applying this to the bilinear form

$$\begin{aligned}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \overline{\alpha_j} \Phi(x_i - x_j) &= \sum_{i=1}^N \sum_{j=1}^N \alpha_i \overline{\alpha_j} (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \widehat{\Phi}(\omega) e^{i(x_i - x_j)^T\omega} d\omega \\ &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \widehat{\Phi}(\omega) \left| \sum_{i=1}^N \alpha_i e^{ix_i^T\omega} \right|^2 d\omega\end{aligned}$$

shows that Φ is positive definite if $\widehat{\Phi}$ is non-negative. However, to prove the second implication, one needs to replace the Lebesgue measure with a more general Borel measure. The entire proof can be found in [16]. One can deduce the following theorem from the proof.

Theorem 2.4. *A positive definite function Φ is strictly positive definite if the carrier of the measure μ from characterization (2.1) contains an open subset.*

Then, we can conclude:

Corollary 2.5. *Let $f \in L_1(\mathbb{R}^d)$ be continuous, non-negative and non-vanishing. Then,*

$$\Phi(x) := \int_{\mathbb{R}^d} f(\omega) e^{-ix^T\omega} d\omega, \quad x \in \mathbb{R}^d \tag{2.2}$$

is strictly positive definite.

Proof. Consider the Borel measure μ defined by

$$\mu(B) := \int_B f(x) d(x).$$

Since f is continuous, non-negative and non-vanishing, its support must contain an interior point. Yet, by definition of μ, the carrier of μ is equal to the support of f and must thus contain an open subset. By theorem 2.4, $\widehat{\mu}$ and hence Φ is strictly positive definite. □

We can now use these results to show that the Gaussian kernels defined by

$$K_\Phi(x, y) := \Phi(x - y) := e^{-\alpha\|x-y\|_2^2}, \quad x, y \in \mathbb{R}^d$$

are strictly positive definite for any shape parameter $\alpha > 0$ on any $\mathbb{R}^d$.

Theorem 2.6. *The Gaussians $\Phi(x) := e^{-\alpha\|x\|_2^2}$ are strictly positive definite on every $\mathbb{R}^d$ for any shape parameter $\alpha > 0$.*

Proof. Let $d \in \mathbb{N}$ and $\alpha > 0$. Define

$$G(x) := e^{-\|x\|_2^2/2}$$

and

$$G_\beta(x) = G(x/\beta)$$

for $x \in \mathbb{R}^d$ and $\beta > 0$. We will see that Φ is strictly positive definite by using G and G_β to write Φ in the form of (2.2).

Firstly, note that $G = \widehat{G}$ holds:

$$\begin{aligned}
\widehat{G}(x) &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} e^{-\|\omega\|_2^2/2} e^{-ix^T\omega} d\omega \\
&= e^{-\|x\|_2^2/2} (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} e^{-\|\omega+ix\|_2^2/2} d\omega \\
&= e^{-\|x\|_2^2/2} \\
&= G(x).
\end{aligned}$$

Moreover, we see that

$$\begin{aligned}
\widehat{G_\beta}(x) &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} G(\omega/\beta) e^{-ix^T\omega} d\omega \\
&= \beta^d (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} G(z) e^{-ix^T\beta z} dz \\
&= \beta^d (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} G(z) e^{-i\beta x^T z} dz \\
&= \beta^d \widehat{G}(\beta x).
\end{aligned}$$

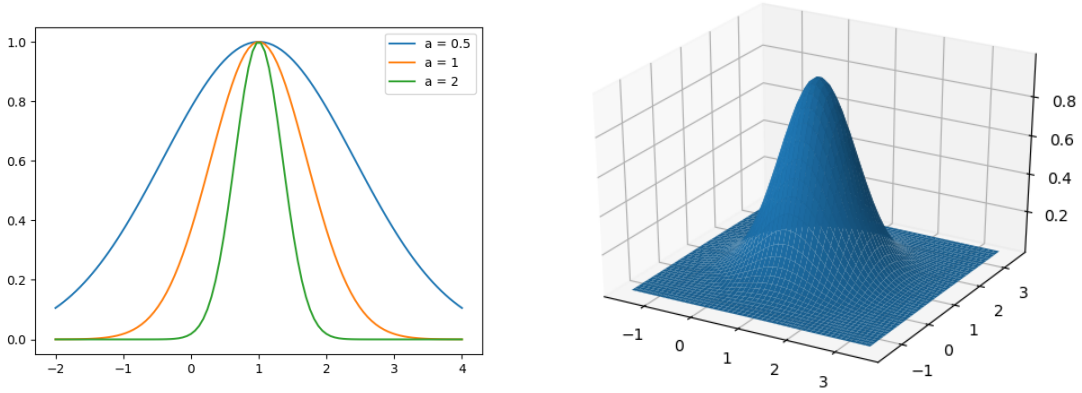

Figure 2.1.: Gaussian Kernel Translates for Shaping Parameter $\alpha = 0.5, 1, 2$ at $x = 1$ (left) and for $\alpha = 1$ at $x = (1, 1)$ (right)

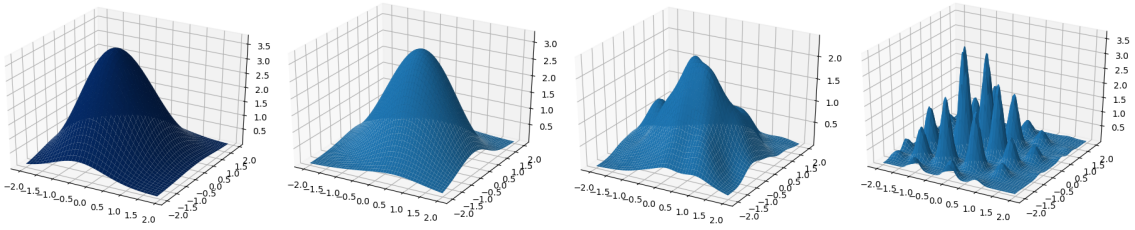

Figure 2.2.: Interpolation with Gaussian Kernel of Target Function (left) for Different Shaping Parameters $\alpha = 1, 2, 5$

Now, let's consider Φ. We can see that

$$\Phi(x) = e^{-\alpha\|x\|_2^2} = G(\sqrt{2\alpha}x) = \widehat{G}(\sqrt{2\alpha}x) = (\sqrt{2\alpha})^{-d}\widehat{G_{\sqrt{2\alpha}}}(x)$$

and hence

$$\Phi(x) = (\sqrt{2\alpha})^{-d}(2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} e^{-\|\omega\|^2/4\alpha} e^{-ix^T\omega} d\omega$$

holds. Since $\omega \mapsto e^{-\|\omega\|^2/4\alpha}$ is integrable, non-negative and non-vanishing, Φ is strictly positive definite. □

Since $\Phi(x) := e^{-\alpha\|x\|_2^2}$ is even and real-valued for any shape parameter $\alpha > 0$ on $\mathbb{R}^d$, the Gaussian kernels K_Φ defined by

$$K_\Phi(x, y) = e^{-\alpha\|x-y\|_2^2}, \quad x, yin\mathbb{R}^d$$

are strictly positive definite by the remark on definition 2.2. When performing kernel interpolation, we will choose certain $x \in \mathbb{R}^d$ and use the functions

$$\begin{aligned} K_\Phi(x, \cdot) : \mathbb{R}^d &\to \mathbb{R} \\ y &\mapsto K_\Phi(x, y) \end{aligned}$$

as basis functions. We will call these functions *kernel translates.* In figure 2.1 some Gaussian kernel translates $K_\Phi(x,\cdot)$ for $\mathbb{R}$ and $\mathbb{R}^2$ and different shaping parameters α are depicted. The choice of the shaping parameter α does make a difference in the interpolation process as figure 2.2 illustrates. It controls the trade off between smoothness and approximation.

Now, we will shortly introduce a second class of radial functions $\Phi = \phi(\|\cdot\|_2)$ that induce the so called *Wendland Kernels* K_Φ. For a thorough derivation of those kernels we refer to [16].

In interpolation theory, locally supported basis functions play an important role as they allow a sparse interpolation matrix and a fast evaluation of the interpolant. An example for the one dimensional space are B-splines. However, the Gaussian kernels that we discussed earlier are globally supported. That's why we are trying to construct a strictly positive definite function $\Phi = \phi(\|\cdot\|)$ on $\mathbb{R}^d$ of the form

$$\phi(x) = \begin{cases} p(x) & \text{for } 0 \le x \le 1 \\ 0 & \text{for } x > 1, \end{cases}$$

where p is a polynomial of minimal degree such that Φ meets a desired smoothness. We will now give an iterative scheme to compute the polynomial p with respect to a given space dimension d and a given smoothness $2k$ of Φ. The proof of this theorem can be found in chapter 9 of [16].

Theorem 2.7. *Let $d, k \in \mathbb{N}$. For $r \in \mathbb{R}^d$, consider the recursively defined polynomial*

$$p_{d,k}(r) = \sum_{j=0}^{\ell+2k} d_{j,k}^{(\ell)} r^j$$

where $\ell = \left\lfloor \frac{d}{2} \right\rfloor + k + 1$ and the coefficients are defined for $0 \le s \le k-1$ by

$$\begin{aligned} d_{j,0}^{(\ell)} &= (-1)^j \binom{\ell}{j} && 0 \le j \le \ell, \\ d_{0,s+1}^{(\ell)} &= \sum_{j=0}^{\ell+2s} \frac{d_{j,s}^{\ell}}{j+2}, \quad d_{1,s+1}^{(\ell)} = 0, \\ d_{j,s+1}^{(\ell)} &= -\frac{d_{j-2,s}^{(\ell)}}{j}, && 2 \le j \le \ell + 2s + 2. \end{aligned}$$

Then, $p_{d,k}$ is a polynomial of degree $\lfloor d/2 \rfloor + 3k + 1$. This the minimal degree to ensure that the functions $\Phi_{d,k} = \phi_{d,k}(\|\cdot\|_2)$ with

$$\phi_{d,k}(r) := \begin{cases} p_{d,k}(r) & 0 \le r \le 1 \\ 0 & r > 1 \end{cases}$$

possess continuous derivatives up to order $2k$. The polynomial $p_{d,k}$ is uniquely defined up to a positive constant factor.

The functions $\phi_{d,k}$ defined by the upper theorem are called *Wendland functions* and induce the so called *Wendland kernels* $K_{\Phi_{d,k}} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined by

$$K_{\Phi_{d,k}}(x,y) := \phi_{d,k}(\|x-y\|_2), \quad x, y \in \mathbb{R}^d.$$

Space dimension	Function	Smoothness
$d = 1$	$\phi_{1,0}(r) = (1-r)_+$	C^0
	$\phi_{1,1}(r) \doteq (1-r)_+^3(3r+1)$	C^2
	$\phi_{1,2}(r) \doteq (1-r)_+^5(8r^2+5r+1)$	C^4
$d \leq 3$	$\phi_{3,0}(r) = (1-r)_+^2$	C^0
	$\phi_{3,1}(r) \doteq (1-r)_+^4(4r+1)$	C^2
	$\phi_{3,2}(r) \doteq (1-r)_+^6(35r^2+18r+3)$	C^4
	$\phi_{3,3}(r) \doteq (1-r)_+^8(32r^3+25r^2+8r+1)$	C^6
$d \leq 5$	$\phi_{5,0}(r) = (1-r)_+^3$	C^0
	$\phi_{5,1}(r) \doteq (1-r)_+^5(5r+1)$	C^2
	$\phi_{5,2}(r) \doteq (1-r)_+^7(16r^2+7r+1)$	C^4

Table 2.1.: Wendland Functions $\phi_{d,k}$

From theorem 6.20 in [16], it follows that the Wendland kernels are strictly positive definite. The next theorem gives a characterization for the Wendland functions in the most important cases.

Theorem 2.8. *For $k = 0, 1, 2, 3$, the Wendland functions $\phi_{d,k}$ have the following form:*

$$\phi_{d,0}(r) = (1-r)_+^{\lfloor d/2 \rfloor + 1}$$
$$\phi_{d,1}(r) \doteq (1-r)_+^{\ell+1}\Big((\ell+1)r+1\Big)$$
$$\phi_{d,2}(r) \doteq (1-r)_+^{\ell+2}\Big((\ell^2+4\ell+3)r2^+(3\ell+6)r+3\Big)$$
$$\phi_{d,3}(r) \doteq (1-r)_+^{\ell+3}\Big((\ell^3+9\ell^2+23\ell+15)r^3+(6\ell^2+36\ell+45)r^2+(15\ell+45)r+15\Big)$$

where $\ell := \lfloor d/2 \rfloor + k + 1$ and $r \in \mathbb{R}^d$. Here, $\doteq$ denotes equality up to a positive constant and

$$(x)_+ := \begin{cases} x & \text{if } x > 0 \\ 0 & \text{else} \end{cases}.$$

In table 2.1, the Wendland functions are explicitly computed for the most important cases and in figure 2.3 some Wendland kernel translates $K_{\Phi_{d,k}}$ are depicted for space dimension $d = 1$ and $d = 2$.

2.2. Reproducing Kernels and Reproducing Kernel Hilbert Spaces

After we discussed some examples of kernels, we will now discuss further theory behind them. As mentioned before, we are looking for a certain class of function Hilbert spaces to perform our interpolation in. In this section we will establish these spaces, the so called *reproducing kernel Hilbert spaces*.

Firstly, recall that a $\mathbb{K}$-Hilbert space $\mathcal{H}$ is isometrically isomorphic to its dual space

$$\mathcal{H}' := \{\phi : \mathcal{H} \to \mathbb{K} \mid \phi \text{ continuous linear functional on } \mathcal{H}\}.$$

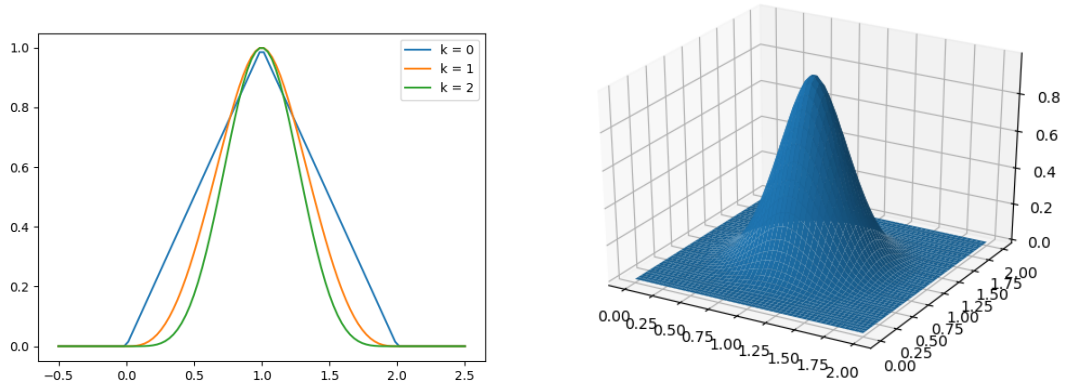

Figure 2.3.: Wendland Kernel Translates for $k = 0, 1, 2$ at $x = 1$ (left) and $k = 1$ at $x = (1, 1)$ (right)

A proof of this theorem can be found in the appendix.

Theorem 2.9 (Fréchet-Riesz theorem)**.** *Let $\mathcal{H}$ be a Hilbert space and let $\phi \in \mathcal{H}'$ be any continuous linear functional on $\mathcal{H}$. Then there exists a unique $x_0 \in \mathcal{H}$ such that $\phi = \langle \cdot, x_0 \rangle$ and $\|\phi\| = \|x_0\|$ holds. Thus, there exits an isometric isomorphism between $\mathcal{H}$ and its dual space $\mathcal{H}'$:*

$$\mathcal{F} : \mathcal{H}' \to \mathcal{H}, \ \phi \mapsto x_0.$$

First of all, let us define a reproducing kernel Hilbert space and explore some basic properties.

Definition 2.10. *Let $\Omega \subset \mathbb{R}^d$. A kernel $K : \Omega \times \Omega \to \mathbb{R}$ is called a reproducing kernel of a Hilbert space $\mathcal{H}$ of functions $f : \Omega \to \mathbb{R}$ if it satisfies the following properties:*

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

If a Hilbert space $\mathcal{H}$ of functions on Ω possesses such a reproducing kernel, it is called a reproducing kernel Hilbert space (RKHS) and denoted by $(\mathcal{H}_K(\Omega), \langle \cdot, \cdot \rangle_{\mathcal{H}_K(\Omega)})$.

When there is no ambiguity, we will omit the subscript and write $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$ for the RKHS inner product and its associated norm.

Remark. The notation $\mathcal{H}_K(\Omega)$ for a RKHS is unambiguous since the reproducing kernel of such a Hilbert space is uniquely defined:

Consider another reproducing kernel K_2 for the RKHS $\mathcal{H}_{K_1}(\Omega)$. Then, property (2), the reproducing property, yields for any $f \in \mathcal{H}_{K_1}(\Omega)$

$$0 = f(x) - f(x) = \langle f, K_1(\cdot, x) - K_2(\cdot, x) \rangle \quad \forall x \in \Omega.$$

Setting $f = K_1(\cdot, x_0) \in \mathcal{H}_{K_1}(\Omega)$ for any fixed $x_0 \in \Omega$ and applying the reproducing property once again yields

$$0 = \langle K_1(\cdot, x) - K_2(\cdot, x), K_1(\cdot, x_0) \rangle = K_1(x_0, x) - K_2(x_0, x) \quad \forall x \in \Omega$$

and $K_1 = K_2$ follows.

Theorem 2.11. *Let $\mathcal{H}$ be a Hilbert space of functions $f : \Omega \to \mathbb{R}$. Then, $\mathcal{H}$ is a RKHS $\mathcal{H} = \mathcal{H}_K(\Omega)$ for a reproducing kernel $K : \Omega \times \Omega \to \mathbb{R}$ if and only if the point evaluation functionals δ_x, defined by*

$$\delta_x : \mathcal{H} \to \mathbb{R}, \quad f \mapsto f(x)$$

for $x \in \Omega$, are continuous.

Proof. First, let K be a reproducing kernel for $\mathcal{H} = \mathcal{H}_K(\Omega)$. Then, by the reproducing property

$$\delta_x(f) = f(x) = \langle f, K(\cdot, x)\rangle \quad \forall f \in \mathcal{H}$$

holds for all $x \in \Omega$. As the inner product and K are both continuous, so is δ_x and $\delta_x \in \mathcal{H}'$ follows for all $x \in \Omega$.

Now, let $\delta_x \in \mathcal{H}'$ for all $x \in \Omega$. By the Fréchet-Riesz theorem (theorem 2.9), there exists an element $f_y \in \mathcal{H}$ for every $y \in \Omega$ such that $\delta_y = \langle \cdot, f_y\rangle$. Hence,

$$f(y) = \langle f, f_y\rangle \quad \forall f \in \mathcal{H}.$$

We can now define the reproducing kernel as the Riesz representer

$$K(\cdot, y) := f_y \in \mathcal{H}$$

for all $y \in \Omega$. Both properties for a reproducing kernel are then assured. □

Proposition 2.12. *Let $\mathcal{H} := \mathcal{H}_K(\Omega)$ be a RKHS of functions $f : \Omega \to \mathbb{R}$. Its reproducing kernel $K : \Omega \times \Omega \to \mathbb{R}$ satisfies the following properties:*

1. *For $x, y \in \Omega$ and their point evaluation functionals $\delta_x, \delta_y \in \mathcal{H}'$, we have*

$$K(x, y) = \langle K(\cdot, x), K(\cdot, y)\rangle_{\mathcal{H}} = \langle \delta_x, \delta_y\rangle_{\mathcal{H}'}.$$

2. *K is symmetric.*
3. *For given $f_n, f \in \mathcal{H}_K(\Omega)$ such that f_n converges to f in RKHS norm, f_n converges also pointwise to f.*

Proof. Applying the Riesz representation $\mathcal{F} : \mathcal{H}' \to \mathcal{H}$ to the point evaluations δ_x for $x \in \Omega$ yields $\mathcal{F}(\delta_x) = K(\cdot, x)$ since

$$\delta_x(f) = f(x) = \langle f, K(\cdot, x)\rangle \quad \forall f \in \mathcal{H}_K(\Omega)$$

and hence

$$\delta_x = \langle \cdot, K(\cdot, x)\rangle$$

which uniquely defines $\mathcal{F}(\delta_x)$. Therefore, for $x, y \in \Omega$ we can use the isometric property of the Riesz isomorphism and obtain

$$\langle \delta_x, \delta_y\rangle_{\mathcal{H}'} = \langle \mathcal{F}(\delta_x), \mathcal{F}(\delta_y)\rangle_{\mathcal{H}} = \langle K(\cdot, x), K(\cdot, y)\rangle_{\mathcal{H}}.$$

Since

$$\langle K(\cdot, x), K(\cdot, y)\rangle_{\mathcal{H}} = \langle K(\cdot, y), K(\cdot, x)\rangle_{\mathcal{H}} = K(x, y)$$

by the reproducing property, the first property follows. From the symmetry of $\langle \cdot, \cdot \rangle$, the symmetry of K

$$K(x, y) = K(y, x) \quad \forall x, y \in \Omega$$

follows immediately. So let $f_n, f \in \mathcal{H}_K(\Omega)$ be such that $\lim_{n\to\infty} \|f_n - f\| = 0$. By the Cauchy Schwartz inequality,

$$|f_n(x) - f(x)| = |\langle f_n - f, K(\cdot, x)\rangle| \leq \|f_n - f\| \, \|K(\cdot, x)\|$$

follows and thus $\lim_{n\to\infty} |f_n(x) - f(x)| = 0$. □

Theorem 2.13. *The reproducing kernel K of a RKHS $\mathcal{H}_K(\Omega)$ is positive definite. It is strictly positive definite if and only if the point evaluation functionals are linearly independent.*

Proof. Let $N \in \mathbb{N}$ and $x_1, ..., x_N \in \Omega$ pairwise distinct points in Ω. For any coefficients $0 \neq (\alpha_1, ..., \alpha_N) \in \mathbb{R}$, the first property in proposition 2.12 yields

$$\sum_{i,j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) = \langle \sum_{i=1}^{N} \alpha_i \delta_{x_i}, \sum_{j=1}^{N} \alpha_j \delta_{x_j} \rangle_{\mathcal{H}_K(\Omega)'} = \left\| \sum_{i=1}^{N} \alpha_i \delta_{x_i} \right\|^2_{\mathcal{H}_K(\Omega)'} \geq 0$$

□

where equality is obtained if and only if the δ_{x_i}, $i = 1, ..., N$, are linearly dependent.

Hence, RKHSs always yield a symmetric (strictly) positive definite kernel. However, in our interpolation process, one usually does not choose the function space beforehand but starts off with a strictly positive definite kernel and hopes that it entails a suitable RKHS to work with. As we will see in the next section, this is fortunately the case for any strictly positive definite kernel.

2.3. Native Spaces for Strictly Positive Definite Kernels

For this section, fix a strictly positive definite kernel $K : \Omega \times \Omega \to \mathbb{R}$. We will now see that we can construct an associated RKHS $\mathcal{H}_K(\Omega)$ with K as its reproducing kernel. This space is called the *Native Space* of K and denoted by $\mathcal{N}_K(\Omega)$.

First, note that for any RKHS $\mathcal{H}_K(\Omega)$ corresponding to K and for all $\alpha_1, ..., \alpha_n \in \mathbb{R}$, $x_1, ..., x_n \in \Omega$

$$\sum_{i=1}^{n} \alpha_i K(\cdot, x_i) \in \mathcal{H}_K(\Omega)$$

holds since K is a reproducing kernel and therefore $K(\cdot, x) \in \mathcal{H}_K(\Omega)$ holds for all $x \in \Omega$ by the reproducing kernel's properties. Hence, the Native Space $\mathcal{N}_K(\Omega)$ of K needs to satisfy that

$$V(X_n) := \operatorname{span}\{K(\cdot, x_1), ..., K(\cdot, x_n)\}$$

is a subspace of $\mathcal{N}_K(\Omega)$ for all $n \in \mathbb{N}$ and all sets $X_n := \{x_1, ..., x_n\}$ of n pairwise distinct points in Ω. Therefore, the completion of

$$N_K(\Omega) := \operatorname{span}\{K(\cdot, x) \mid x \in \Omega\}$$

will be our candidate for $\mathcal{N}_K(\Omega)$. Moreover, we can see that for any RKHS $\mathcal{H}_K(\Omega)$ corresponding to K and $f = \sum_{i=1}^n \alpha_i K(\cdot, x_i) \in \mathcal{H}_K(\Omega)$, the norm of f can be computed by

$$\|f\|^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle K(\cdot, x_i), K(\cdot, x_j)\rangle_{\mathcal{H}_K(\Omega)} = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j) \tag{2.3}$$

by the first property in proposition 2.12. To find a candidate for the inner product of $\mathcal{N}_K(\Omega)$, we will thus equip $N_K(\Omega)$ with the bilinear form

$$\langle \sum_{i=1}^n \alpha_i K(\cdot, x_i), \sum_{j=1}^m \beta_j K(\cdot, y_j)\rangle_K := \sum_{i=1}^n \sum_{j=1}^m \alpha_i \beta_j K(x_i, y_j) \quad x_1, ..., x_n, y_1, ..., y_m \in \Omega.$$

Theorem 2.14. *$(N_K(\Omega), \langle \cdot, \cdot \rangle_K)$ defines a pre-Hilbert space and K is a reproducing kernel to $N_K(\Omega)$.*

Proof. By construction, $\langle \cdot, \cdot \rangle_K$ is bilinear and since K is symmetric, also $\langle \cdot, \cdot \rangle_K$ is symmetric. Let $0 \neq f = \sum_{i=1}^n \alpha_i K(\cdot, x_i) \in N_K(\Omega)$. Then,

$$\|f\|_K^2 = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j K(x_i, x_j) > 0$$

holds since K is strictly positive definite. Therefore, $\langle \cdot, \cdot \rangle_K$ is also positive definite and hence an inner product on $N_K(\Omega)$. Moreover,

$$K(\cdot, x) \in N_K(\Omega)$$

obviously holds for all $x \in \Omega$ and for any $f = \sum_{i=1}^n \alpha_i K(\cdot, x_i) \in N_K(\Omega)$ and $x \in \Omega$ we have

$$\langle f, K(\cdot, x)\rangle_K = \sum_{i=1}^n \alpha_i \langle K(\cdot, x_i), K(\cdot, x)\rangle_K = \sum_{i=1}^n \alpha_i K(x_i, x) = f(x),$$

which proves the reproducing properties for K. □

We will thus define the native space of K as follows.

Definition 2.15. *The corresponding native space $\mathcal{N}_K(\Omega)$ to K is defined as the completion*

$$\mathcal{N}_K(\Omega) := \overline{N_K(\Omega)}^{\langle \cdot, \cdot \rangle_K}$$

equipped with the inner product induced by $\langle \cdot, \cdot \rangle_K$.

In the next theorem we will see that this definition is powerful to the extent that the native space of K is uniquely defined.

Theorem 2.16. *Let $\mathcal{H}_K(\Omega)$ be a RKHS such that K is its reproducing kernel. Then, the Hilbert spaces $\mathcal{H}_K(\Omega)$ and $\mathcal{N}_K(\Omega)$ and their inner products coincide.*

Proof. First, we will show that $\mathcal{N}_K(\Omega) \subset \mathcal{H}_K(\Omega)$. As we discussed above, $N_K(\Omega)$ is a subspace of both $\mathcal{H}_K(\Omega)$ and $\mathcal{N}_K(\Omega)$. By consideration (2.3) the norms coincide on $N_K(\Omega)$. Let $f \in \mathcal{N}_K(\Omega)$. Since $N_K(\Omega)$ is dense in $\mathcal{N}_K(\Omega)$, there exists a Cauchy sequence $(f_n)_n$ in $N_K(\Omega)$ that converges to f with respect to $\|\cdot\|_{\mathcal{N}_K(\Omega)}$. By the third property in proposition 2.12, we even have pointwise convergence

$$f(x) = \lim_{n\to\infty} f_n(x) \quad \forall x \in \Omega.$$

However, since $N_K(\Omega)$ is also a subspace of $\mathcal{H}_K(\Omega)$ and their norms coincide, $(f_n)_n$ is also a Cauchy sequence in $\mathcal{H}_K(\Omega)$. Therefore, there exists a $g \in \mathcal{H}_K(\Omega)$ such that f_n converges to g in $\mathcal{H}_K(\Omega)$-norm. Yet, by the reproducing property of K we get

$$g(x) = \langle g, K(\cdot, x)\rangle_{\mathcal{H}_K(\Omega)} = \lim_{n\to\infty} \langle f_n, K(\cdot, x)\rangle_{\mathcal{H}_K(\Omega)} = \lim_{n\to\infty} f_n(x) = f(x)$$

for all $x \in \Omega$ and since $\|f_n\|_{\mathcal{H}_K(\Omega)} = \|f_n\|_{\mathcal{N}_K(\Omega)}$ holds for all $n \in \mathbb{N}$, also $\|f\|_{\mathcal{N}_K(\Omega)} = \|g\|_{\mathcal{H}_K(\Omega)}$ follows and hence, the inner products coincide.
Now, assume that $\mathcal{N}_K(\Omega) \neq \mathcal{H}_K(\Omega)$. Then, the orthogonal complement of $\mathcal{N}_K(\Omega)$ in $\mathcal{H}_K(\Omega)$ is not the null space and there exists a $0 \neq g \in \mathcal{N}_K(\Omega)^\perp$. Since

$$\langle g, f\rangle = 0 \quad \forall f \in \mathcal{N}_K(\Omega)$$

holds,

$$0 = \langle g, K(\cdot, x)\rangle = g(x) \quad \forall x \in \Omega$$

follows which contradicts the assumption that $g \neq 0$. □

Now, let's consider the specific case that $\Omega = \mathbb{R}^d$ and K_Φ is a strictly positive definite, translational invariant kernel, i.e. a kernel defined by

$$K_\Phi(x, y) := \Phi(x - y)$$

for a fixed $\Phi : \mathbb{R}^d \to \mathbb{R}$ such as the Gaussian and Wendland kernel for example. In this case, we can use the previous uniqueness property to discover that the native space of K_Φ is the space of smooth functions on $\mathbb{R}^d$ under some condition.

Theorem 2.17. *Let $\Phi \in \mathcal{C}(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$ be real-valued and strictly positive definite. Define*

$$\mathcal{G} := \{f \in L_2(\mathbb{R}^d) \cap \mathcal{C}(\mathbb{R}^d) \mid \widehat{f}\Big/\sqrt{\widehat{\Phi}} \in L_2(\mathbb{R}^d)\}$$

and equip it with the bilinear form

$$\begin{aligned}\langle f, g\rangle_{\mathcal{G}} :&= (2\pi)^{-d/2}\langle \widehat{f}/\sqrt{\widehat{\Phi}}, \widehat{g}/\sqrt{\widehat{\Phi}}\rangle_{L_2(\mathbb{R}^d)} \\ &= (2\pi)^{-d/2} \int_{\mathbb{R}^d} \frac{\widehat{f}(\omega)\overline{\widehat{g}(\omega)}}{\widehat{\Phi}(\omega)} d\omega.\end{aligned}$$

Then, $(\mathcal{G}, \langle\cdot,\cdot\rangle_{\mathcal{G}})$ is a real Hilbert space with reproducing kernel K_Φ.

Proof. Firstly we will show that $\langle \cdot, \cdot \rangle_{\mathcal{G}}$ is a real-valued inner product on $\mathcal{G}$ and hence that $(\mathcal{G}, \langle \cdot, \cdot \rangle_{\mathcal{G}})$ is a pre-Hilbert space.

Note that for any real-valued f the Fourier transform satisfies

$$\overline{\widehat{f}(x)} = (2\pi)^{-d/2} \int_{\mathbb{R}^d} \overline{f(\omega)} e^{-i(-x)^T \omega} d\omega = \widehat{f}(-x). \tag{2.4}$$

This thus holds for the L_2-Fourier transform almost everywhere. We can use this to show that $\langle \cdot, \cdot \rangle_{\mathcal{G}}$ is real-valued. Note that since Φ is real-valued and symmetric, $\widehat{\Phi}$ is real-valued as well:

$$\overline{\widehat{\Phi}(\omega)} = \widehat{\Phi}(-\omega) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \Phi(z) e^{-i(-\omega)^T z} dz = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \Phi(-z) e^{-i\omega^T(-z)} dz = \widehat{\Phi}(\omega).$$

For $f, g \in \mathcal{G}$ follows

$$\begin{aligned}
\int_{\mathbb{R}^d} \frac{\widehat{f}(\omega)\overline{\widehat{g}(\omega)}}{\widehat{\Phi}(\omega)} d\omega &= \int_{\omega_1 > 0} \frac{\widehat{f}(\omega)\overline{\widehat{g}(\omega)}}{\widehat{\Phi}(\omega)} + \frac{\widehat{f}(-\omega)\overline{\widehat{g}(-\omega)}}{\widehat{\Phi}(-\omega)} d\omega \\
&= \int_{\omega_1 > 0} \frac{\widehat{f}(\omega)\overline{\widehat{g}(\omega)} + \overline{\widehat{f}(\omega)}\widehat{g}(\omega)}{\widehat{\Phi}(\omega)} d\omega \\
&= 2 \int_{\omega_1 > 0} \frac{\mathrm{Re}(\widehat{f}(\omega)\overline{\widehat{g}(\omega)})}{\widehat{\Phi}(\omega)} d\omega \in \mathbb{R}.
\end{aligned}$$

The bilinear form $\langle \cdot, \cdot \rangle_{\mathcal{G}}$ is hence real-valued. It follows directly from the definition that $\langle \cdot, \cdot \rangle_{\mathcal{G}}$ is $\mathbb{R}$-bilinear. That it is symmetric follows from the conjugate symmetry of $\langle \cdot, \cdot \rangle_{L_2(\mathbb{R}^d)}$ and real values. And positive definiteness follows from positive definiteness of $\langle \cdot, \cdot \rangle_{L_2(\mathbb{R}^d)}$. Hence, $\langle \cdot, \cdot \rangle_{\mathcal{G}}$ is a real inner product on $\mathcal{G}$.

Next, we will prove that $(\mathcal{G}, \langle \cdot, \cdot \rangle_{\mathcal{G}})$ is a Hilbert space. We thus need to prove completeness. Take any Cauchy sequence $(f_n)_{n \in \mathbb{N}}$ of $(\mathcal{G}, \langle \cdot, \cdot \rangle_{\mathcal{G}})$. Since

$$\left\| \widehat{f_n}/\sqrt{\widehat{\Phi}} - \widehat{f_m}/\sqrt{\widehat{\Phi}} \right\|^2_{L_2(\mathbb{R}^d)} = (2\pi)^{d/2} \left\| f_n - f_m \right\|^2_{\mathcal{G}},$$

$(\widehat{f_n}/\sqrt{\widehat{\Phi}})_n$ is a Cauchy sequence in $L_2(\mathbb{R}^d)$. Since $L_2(\mathbb{R}^d)$ is complete, there exists a limit $g \in L_2(\mathbb{R}^d)$ to $(\widehat{f_n}/\sqrt{\widehat{\Phi}})_n$. Consider $g\sqrt{\widehat{\Phi}}$. By Hölder's inequality we have

$$\begin{aligned}
\int_{\mathbb{R}^d} |g(\omega)\sqrt{\widehat{\Phi}(\omega)}| d\omega &\leq \left(\int_{\mathbb{R}^d} |g(\omega)|^2 d\omega \right)^{\frac{1}{2}} \left(\int_{\mathbb{R}^d} |\widehat{\Phi}(\omega)| d\omega \right)^{\frac{1}{2}} \\
&= \|g\|_{L_2(\mathbb{R}^d)} \|\widehat{\Phi}\|^{\frac{1}{2}}_{L_1(\mathbb{R}^d)}.
\end{aligned}$$

In chapter 6 of [16], various properties for positive definite functions were shown, including the fact that $\widehat{\Phi} \in L_1(\mathbb{R}^d)$ if $\Phi \in \mathcal{C}(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$ is positive definite (see Corollary 6.12 in [16]). Hence, $g\sqrt{\widehat{\Phi}} \in L_1(\mathbb{R}^d)$ as $\widehat{\Phi} \in L_1(\mathbb{R}^d)$ by [16] and $g \in L_2(\mathbb{R}^d)$. We can even show that $g\sqrt{\widehat{\Phi}} \in L_1(\mathbb{R}^d) \cap L_2(\mathbb{R}^d)$. Since any positive definite function is bounded (see Theorem 6.2 in [16]), Φ and therefore also $\widehat{\Phi}$ is bounded. Moreover,

$$\int_{\mathbb{R}^d} |g(\omega)\sqrt{\widehat{\Phi}(\omega)}|^2 d\omega \leq \|\widehat{\Phi}\|_{L_\infty(\mathbb{R}^d)} \|g\|^2_{L_2(\mathbb{R}^d)} < \infty$$

and $g\sqrt{\widehat{\Phi}} \in L_2(\mathbb{R}^d)$. Plancherel's theorem then allows us to define

$$\begin{aligned} f(x) :&= (g\sqrt{\widehat{\Phi}})^\vee(x) \\ &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} g(\omega)\sqrt{\widehat{\Phi}(\omega)} e^{ix^T\omega} d\omega, \quad x \in \mathbb{R}^d \end{aligned}$$

and provides a continuous $f \in L_2(\mathbb{R}^d)$ that satisfies $\widehat{f} = g\sqrt{\widehat{\Phi}}$. This will be our candidate for the limit of $(f_n)_n$ in $\mathcal{G}$. In order to see that $f \in \mathcal{G}$, it remains to show that f is real-valued. From writing f and f_n in terms of its Fourier transforms and applying Hölder's inequality

$$\begin{aligned} |f(x) - f_n(x)| &\leq (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} |g(\omega)\sqrt{\widehat{\Phi}(\omega)} - \widehat{f_n}(\omega)| d\omega \\ &\leq (2\pi)^{-\frac{d}{2}} \left\|\widehat{\Phi}\right\|_{L_1(\mathbb{R}^d)} \left\| g - \widehat{f_n}/\sqrt{\widehat{\Phi}} \right\|_{L_2(\mathbb{R}^d)} \underset{n\to\infty}{\longrightarrow} 0 \end{aligned}$$

and that $f_n(x) \in \mathbb{R}$, it follows that $f(x) \in \mathbb{R}$ for $x \in \mathbb{R}^d$. Thus, $f \in \mathcal{G}$. It remains to check that $\|f_n - f\|_{\mathcal{G}} \underset{n\to\infty}{\longrightarrow} 0$. This however follows directly from the definition of f:

$$\|f_n - f\|_{\mathcal{G}}^2 = (2\pi)^{-\frac{d}{2}} \left\| \widehat{f_n}/\sqrt{\widehat{\Phi}} - g \right\|_{L_2(\mathbb{R}^d)}^2 \underset{n\to\infty}{\longrightarrow} 0.$$

Therefore, $(\mathcal{G}, \langle\cdot,\cdot\rangle_{\mathcal{G}})$ is complete and thus a real Hilbert space.

Lastly, we will prove that K_Φ is a reproducing kernel for $(\mathcal{G}, \langle\cdot,\cdot\rangle_{\mathcal{G}})$. On the one hand, this requires to show that $K_\Phi(\cdot, y) = \Phi(\cdot - y) \in \mathcal{G}$ for every $y \in \mathbb{R}^d$. Fix $y \in \mathbb{R}^d$. Since Φ is continuous and real-valued, $\Phi(\cdot - y) \in \mathcal{C}(\mathbb{R}^d)$ is real-valued and since Φ is bounded by $\Phi(0)$ (see Corollary 6.2 in [16]) and $\Phi \in L_1(\mathbb{R}^d)$, it follows that

$$\int_{\mathbb{R}^d} |\Phi(\omega - y)|^2 d\omega \leq \Phi(0)\|\Phi\|_{L_1(\mathbb{R}^d)} < \infty$$

and hence that $\Phi(\cdot - y) \in L_2(\mathbb{R}^d)$. And from

$$\Phi(\cdot - y)^\wedge(x) = (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \Phi(\omega) e^{-i(\omega+y)^T x} d\omega = \widehat{\Phi}(x) e^{-iy^T x} \tag{2.5}$$

and hence

$$\begin{aligned} \left\| \Phi(\cdot - y)^\wedge / \sqrt{\widehat{\Phi}} \right\|_{L_2(\mathbb{R}^d)}^2 &= \left\| \sqrt{\widehat{\Phi}(\cdot)} e^{-iy^T\cdot} \right\|_{L_2(\mathbb{R}^d)}^2 \\ &= \int_{\mathbb{R}^d} |\sqrt{\widehat{\Phi}(\omega)} e^{-iy^T\omega}|^2 d\omega \\ &= \|\widehat{\Phi}\|_{L_1(\mathbb{R}^d)} < \infty \end{aligned}$$

it follows that $\Phi(\cdot - y)^\wedge/\sqrt{\widehat{\Phi}} \in L_2(\mathbb{R}^d)$ and hence that $\Phi(\cdot - y) \in \mathcal{G}$. On the other hand, we need to show that $\langle K_\Phi(\cdot, y), f\rangle_{\mathcal{G}} = f(y)$ for every $y \in \mathbb{R}^d$ and $f \in \mathcal{G}$. Let $f \in \mathcal{G}$ and $y \in \mathbb{R}^d$.

Using (2.5) and (2.4) we see that

$$\begin{aligned}\langle \Phi(\cdot - y), f\rangle_{\mathcal{G}} &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \frac{\Phi(\cdot - y)^\wedge(\omega)\overline{\widehat{f}(\omega)}}{\widehat{\Phi}(\omega)} d\omega \\ &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \frac{\widehat{\Phi}(\omega) e^{-i\omega^T y}\overline{\widehat{f}(\omega)}}{\widehat{\Phi}(\omega)} d\omega \\ &= (2\pi)^{-\frac{d}{2}} \int_{\mathbb{R}^d} \widehat{f}(-\omega) e^{-i\omega^T y} d\omega \\ &= f(y)\end{aligned}$$

by recovering f from its Fourier transform. Finally, it follows that $(\mathcal{G}, \langle\cdot,\cdot\rangle_{\mathcal{G}})$ is a RKHS to K_Φ. □

By theorem 2.16, we can deduce that the Native Space $\mathcal{N}_{K_\Phi}(\Omega)$ of K_Φ and its inner product coincides with $(\mathcal{G}, \langle\cdot,\cdot\rangle_{\mathcal{G}})$. In this case, we can consider the native space of K_Φ in a way as a generalized Sobolev space. Remember that a Sobolev space of order s is the space

$$H^s(\mathbb{R}^d) := \{f \in L_2(\mathbb{R}^d) \cap \mathcal{C}(\mathbb{R}^d) \mid \hat{f}(\cdot)(1 + \|\cdot\|_2^2)^{\frac{s}{2}} \in L_2(\mathbb{R}^d)\}.$$

When the Fourier transform of Φ decays only algebraically, then the native space is a Sobolev space:

Corollary 2.18. *Let*

$$c_1(1 + \|\cdot\|_2^2)^{-s} \leq \widehat{\Phi} \leq c_2(1 + \|\cdot\|_2^2)^{-s}$$

for constants $c_1, c_2 > 0$ and $s > \frac{d}{2}$. Then, the Native space $\mathcal{N}_{K_\Phi}(\Omega)$ coincides with the Sobolev space $H^s(\mathbb{R}^d)$ and their inner products are equivalent.

Proof. Let $f \in \mathcal{G} = \mathcal{N}_{K_\Phi}(\mathbb{R}^d)$. Then $\widehat{f}/\sqrt{\widehat{\Phi}} \in L_2(\mathbb{R}^d)$. Since $\hat{\Phi} \leq c_2(1 + \|\cdot\|_2^2)^{-s}$ and

$$\widehat{f}(\omega)(1 + \|\omega\|_2^2)^{\frac{s}{2}} \leq \sqrt{c_2}\widehat{f}(\omega)/\sqrt{\widehat{\Phi}(\omega)},$$

$\widehat{f}(\cdot)(1 + \|\cdot\|_2^2)^{\frac{s}{2}} \in L_2(\mathbb{R}^d)$ and hence $f \in H^s(\mathbb{R}^d)$ follows.

On the other hand, let $f \in H^s(\mathbb{R}^d)$. Then, $\widehat{f}(\cdot)(1+\|\cdot\|_2^2)^{\frac{s}{2}} \in L_2(\mathbb{R}^d)$. Since $c_1(1+\|\cdot\|_2^2)^{-s} \leq \widehat{\Phi}$ and

$$\widehat{f}(\omega)/\sqrt{\widehat{\Phi}(\omega)} \leq \widehat{f}(\omega)(1 + \|\omega\|_2^2)^{\frac{s}{2}}/\sqrt{c_1},$$

$\widehat{f}/\sqrt{\widehat{\Phi}} \in L_2(\mathbb{R}^d)$ and hence $f \in \mathcal{G}$ follows. □

In theorem 10.35 of [16] it is proven that the Fourier transform of the Wendland function $\Phi_{d,k}$, that we introduced in section 2.1, decays algebraically and hence the following corollary follows.

Corollary 2.19. *Let $K_{d,k} = \Phi_{d,k}(\|\cdot\|_2)$ be a Wendland kernel and $\Omega = \mathbb{R}^d$. If $k = 0$, let $d \geq 3$. Then, the native space is the Sobolev space*

$$\mathcal{N}_{K_{d,k}}(\Omega) = H^{\frac{d}{2}+k+\frac{1}{2}}(\mathbb{R}^d).$$

CHAPTER 3

Interpolation Using A Kernel Basis

At this point, we have already studied strictly positive definite kernels and their native spaces. Let us now apply our newly gained skills to an interpolation problem.

First, we will approximate (possibly unknown) target functions with scalar output and see that we can transfer the approach to vector valued functions easily. Then, we will study a measure of the interpolation quality, the *Power Function*, and use it to introduce different greedy algorithms. Lastly, we will study the P-greedy algorithm more thorough.

3.1. Interpolation Procedure for Scalar Valued Functions

Let $\Omega \subset \mathbb{R}^d$ be a set of scattered data of any dimension d and let $f : \Omega \to \mathbb{R}$ be any target function. Also, let $X_N := \{x_1, ..., x_N\} \subset \Omega$ be N pairwise distinct data sites. Our objective is to find a surrogate model $s_f : \Omega \to \mathbb{R}$ that interpolates the given data sites in X_N according to the following interpolation conditions

$$s_f(x_i) = f(x_i) \quad \forall x_i \in X_N. \tag{3.1}$$

As a general interpolation approach we consider s_f to be of the form

$$s_f(x) = \sum_{i=1}^{N} \psi_i(x)\alpha_i$$

for basis functions $\psi_1, ..., \psi_N : \Omega \to \mathbb{R}$ and coefficients $\alpha_1, ..., \alpha_N \in \mathbb{R}$. In kernel interpolation, we fix a strictly positive definite kernel K and choose the basis functions to be the kernel translates

$$\psi_i = K(\cdot, x_i) : \Omega \to \mathbb{R}, \quad i = 1, ..., N$$

of the given data sites. The interpolant can thus be written as

$$s_f(x) = \sum_{i=1}^{N} K(x, x_i)\alpha_i$$

and is an element of the N-dimensional subspace

$$V(X_N) := \operatorname{span}\{K(\cdot, x_1), ..., K(\cdot, x_N)\} \subset \mathcal{N}_K(\Omega)$$

of the underlying Native space $\mathcal{N}_K(\Omega)$ to K. The interpolation problem then reduces to finding a unique solution to the following linear system

$$\begin{pmatrix} K(x_1, x_1) & \dots & K(x_1, x_N) \\ \vdots & & \vdots \\ K(x_N, x_1) & \dots & K(x_N, x_N) \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{pmatrix},$$

or in short terms

$$A_{K,X_N}\alpha = f$$

for the kernel matrix A_{K,X_N}, a column vector $\alpha \in \mathbb{R}^N$ of all coefficients and a column vector $f \in \mathbb{R}^N$ of all function values. Since we chose K to be strictly positive definite, the kernel matrix A_{K,X_N} is positive definite and thus invertible. Therefore, a unique solution α to the linear system is always guaranteed.

If $f \in \mathcal{N}_K(\Omega)$ holds, we can show the following optimality properties for the kernel interpolant. Recall that for any Hilbert space $(H, (\cdot,\cdot))$ the Pythagorean Theorem holds: For $u, v \in H$ orthogonal, i.e. $(u, v) = 0$, the following holds

$$\|u\|^2 = \|u - v\|^2 + 2(u, v) + \|v\|^2 = \|u - v\|^2 + \|v\|^2 .$$

Proposition 3.1. *If $f \in \mathcal{N}_K(\Omega)$, the kernel interpolant s_f of f on X_N is precisely the orthogonal projection of f onto $V(X_N)$ and thus the best approximation of f in $V(X_N)$ with respect to $\|\cdot\|_{\mathcal{N}_K(\Omega)}$.*

Proof. Since $f \in \mathcal{N}_K(\Omega)$, we can apply the kernel's reproducing property to $s_f - f$. Since we can write s_f in terms of the basis $(K(\cdot, x_1), ..., K(\cdot, x_N))$ of $V(X_N)$

$$s_f = \sum_{i=1}^{N} \alpha_i K(\cdot, x_i)$$

and it satisfies

$$\langle s_f - f, K(\cdot, x_j)\rangle = s_f(x_j) - f(x_j) = 0 \quad \text{for } j = 1, ..., N,$$

s_f is the orthogonal projection of f onto $V(X_N)$.

Fix an arbitrary $g \in V(X_N)$. Applying the Pythagorean Theorem to $s_f - g \in V(X_N)$ and $s_f - f \in V(X_N)^\perp$

$$\|f - g\|^2 = \|f - s_f\|^2 + \|s_f - g\|^2 \geq \|s_f - f\|^2$$

shows that s_f is the best approximation of f in $V(X_N)$ with respect to $\|\cdot\|_{\mathcal{N}_K(\Omega)}$. □

Proposition 3.2. *Let $f \in \mathcal{N}_K(\Omega)$. Consider the set of all interpolants of f in $\mathcal{N}_K(\Omega)$*

$$\mathcal{S} := \{s \in \mathcal{N}_K(\Omega) \mid s(x_i) = f(x_i) \quad \forall i = 1, ..., N.$$

The kernel interpolant s_f has minimal $\mathcal{N}_K(\Omega)$-norm over all elements of $\mathcal{S}$, i.e.

$$\|s_f\| \leq \|s\| \quad \forall s \in \mathcal{S}.$$

Proof. Let $s \in \mathcal{S}$. Applying the reproducing property and the fact that $s_f(x_i) = f(x_i) = s(x_i)$ for all $i = 1, ..., N$ yields

$$\langle s_f - s, s_f\rangle = \sum_{i=1}^{N} \alpha_i \langle s_f - s, K(\cdot, x_i)\rangle = \sum_{i=1}^{N} \alpha_i (s_f(x_i) - s(x_i)) = 0.$$

Hence, $s_f - s \perp s_f$. Applying the Pythagorean theorem gives

$$\|s\|^2 = \|s - s_f\|^2 + \|s_f\|^2 \geq \|s_f\|^2 .$$

□

3.2. From Scalar Valued Functions to Vector Valued Functions

The previous considerations can be easily generalized to the case of vector valued functions. Again, let $x_1, ..., x_N$ be pairwise distinct data sites in $\Omega \subset \mathbb{R}^d$ and fix a strictly positive definite kernel K on Ω. Now consider a model $f : \Omega \to \mathbb{R}^q$ with vectorial output of dimension q to be interpolated. Then, the kernel interpolate is of the form

$$s_f(x) = \sum_{i=1}^{N} K(\cdot, x_i)\alpha_i$$

for column vectors $\alpha_1, ..., \alpha_N \in \mathbb{R}^q$ of coefficients. The interpolation conditions can be written as

$$s_f(x_i)^T = f(x_i)^T \quad \forall i = 1, ..., N.$$

This results in the linear system

$$A_{K,X_N}\alpha = f,$$

where

$$\alpha = \begin{pmatrix} \alpha_1^T \\ \vdots \\ \alpha_N^T \end{pmatrix} \in \mathbb{R}^{N\times q} \text{ and } f = \begin{pmatrix} f(x_1)^T \\ \vdots \\ f(x_N)^T \end{pmatrix} \in \mathbb{R}^{N\times q}.$$

This can be seen as solving q interpolation problems with the kernel matrix A_{K,X_N}, one for each component of f. We can transfer our previous analysis by defining the Hilbert space of functions whose components are elements of the underlying Native Space

$$\mathcal{N}_K(\Omega)^q := \{f : \Omega \to \mathbb{R}^q \mid (f)_i \in \mathcal{N}_K(\Omega)\ \forall i = 1, ..., q\}$$

equipped with the inner product

$$\langle f, g\rangle_q := \sum_{i=1}^{q} \langle f_i, g_i\rangle.$$

Hence, the norm of a function $f \in \mathcal{N}_K(\Omega)^q$ can be computed as

$$\|f\|^2_{\mathcal{N}_K(\Omega)^q} = \sum_{i=1}^{q} \left\|f^{(i)}\right\|^2_{\mathcal{N}_K(\Omega)}.$$

We then compute the kernel interpolate for each component individually. However, to avoid computing q independent sets of centers we assume a shared common subspace $V(X_N)$.

In what follows we will work with this generalized interpolation approach.

3.3. The Power Function

We are now introducing the Power Function which will play an important role in the interpolation process. We will give the definition and examine some of its useful properties to develop an interpolation algorithm.

Let $\Omega \subset \mathbb{R}^d$ and $f \in \mathcal{N}_K(\Omega)$. For any set $X_n := \{x_1, ..., x_n\} \subset \Omega$ of n pairwise distinct points in Ω we will denote the subspace of $\mathcal{N}_K(\Omega)$ generated by all kernel translates with centers in X_n by

$$V(X_n) := \text{span}\{K(\cdot, x_1), ..., K(\cdot, x_n)\}$$

and the orthogonal projection operator onto $V(X_n)$ by

$$\Pi_{V(X_n)} : \mathcal{N}_K(\Omega) \to V(X_n).$$

For $n = 0$, we will set $X_0 := \emptyset$ and $V(X_0) := \{0\}$. As we saw in lemma 3.1, $\Pi_{V(X_n)}(f)$ is precisely the interpolant s_f of our kernel interpolation of f on X_n.

Definition 3.3. *The Power Function $P_{V(X_n)} : \mathbb{R}^d \to \mathbb{R}$ measures the interpolation error over all $f \in \mathcal{N}_K(\Omega)$ and is defined as*

$$P_{V(X_n)}(x) := \sup_{f \in \mathcal{N}_K(\Omega)} \frac{|f(x) - \Pi_{V(X_n)}(f)(x)|}{\|f\|}, \quad x \in \mathbb{R}$$

for $n \geq 0$.

From the definition we immediately see that

$$\left| f(x) - \Pi_{V(X_n)}(f)(x) \right| \leq P_{V(X_n)}(x) \, \|f\| \tag{3.2}$$

holds for all $x \in \mathbb{R}^d$, which gives a bound to the interpolation error that decomposes into a factor independent of f and $\|f\|$.

First of all, we will prove a useful characterization of the Power Function.

Lemma 3.4. *For all $V(X_n) \subset \mathcal{N}_K(\Omega)$, $n \geq 0$, and $x \in \Omega$ we have*

$$P_{V(X_n)}(x) = \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\|.$$

Proof. Consider $f \in \mathcal{N}_K(\Omega)$ and any orthonormal basis $(v_1, ..., v_n)$ of $V(X_n)$. Let $x \in \Omega$. Then, the kernel interpolant $s_f = \Pi_{V(X_n)}(f)$ can be written as

$$\Pi_{V(X_n)}(f)(x) = \sum_{i=1}^{N} \langle f, v_i \rangle v_i(x).$$

Writing the interpolant this way and applying the kernel's reproducing property yields

$$\begin{aligned} f(x) - \Pi_{V(X_n)}(f)(x) &= \langle f, K(\cdot, x) \rangle - \langle \sum_{k=1}^{n} \langle f, v_k \rangle v_k, K(\cdot, x) \rangle \\ &= \langle f, K(\cdot, x) \rangle - \sum_{k=1}^{n} \langle f, v_k \rangle \langle v_k, K(\cdot, x) \rangle \\ &= \langle f, K(\cdot, x) \rangle - \langle f, \sum_{k=1}^{n} \langle v_k, K(\cdot, x) \rangle v_k \rangle \end{aligned}$$

and thus

$$f(x) - \Pi_{V(X_n)}(f)(x) = \langle f, K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \rangle. \tag{3.3}$$

Applying the Cauchy Schwartz inequality leads to

$$|f(x) - \Pi_{V(X_n)}(f)(x)| \leq \|f\| \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\|$$

and therefore

$$P_{V(X_n)}(x) = \sup_{f \in \mathcal{N}_K(\Omega)} \frac{|f(x) - \Pi_{V(X_n)}(f)(x)|}{\|f\|} \leq \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\|. \tag{3.4}$$

Now define

$$f_x := \frac{K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x))}{\left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\|}.$$

Since $K(\cdot, x) \in \mathcal{N}_K(\Omega)$ and $\Pi_{V(X_n)}(K(\cdot, x)) \in V(X_n) \subset \mathcal{N}_K(\Omega)$, we have $f_x \in \mathcal{N}_K(\Omega)$. By equation (3.3) we get

$$f_x(x) - \Pi_{V(X_n)}(f_x)(x) = \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\|^2$$

and thus equality in (3.4) is reached. □

Proposition 3.5. *For countably many pairwise distinct $x_i \in \Omega$, the $K(\cdot, x_i)$ are linear independent.*

Proof. Let $x_1, x_2, ... \in \Omega$ be countably many pairwise distinct points in Ω and let $\alpha = (\alpha_1, \alpha_2, ...)$ be countably many real coefficients such that $\sum_{i \in \mathbb{N}} \alpha_i K(\cdot, x_i) = 0$. In particular $\alpha^T A \alpha = \sum_{i \in \mathbb{N}} \alpha_j \sum_{i=1}^{n} \alpha_i K(x_j, x_i) = 0$ where A denotes the kernel matrix. Since A is positive definite, $\alpha = 0$ follows. □

Corollary 3.6. *The Power Function $P_{V(X_n)}$ only vanishes on X_n:*

$$P_{V(X_n)}(x) = 0 \Leftrightarrow x \in X_n.$$

Proof. If $x_i \in X_n$ holds, $P_{V(X_n)}(x_i) = 0$ follows immediately from the interpolation condition $f(x_i) = \Pi_{V(X_n)}(x_i)$.

Let $x \in \Omega$ be such that $P_{V(X_n)}(x) = 0$. From lemma 3.4 follows

$$K(\cdot, x) = \Pi_{V(X_n)}(K(\cdot, x)) \in V(X_n).$$

Since $(K(\cdot, x_1), ..., K(\cdot, x_n), K(\cdot, x))$ are linear independent by proposition 3.5,

$$K(\cdot, x) \in \{K(\cdot, x_1), ..., K(\cdot, x_n)\}$$

follows. □

Remark. Let $\mathcal{H}$ be any separable Hilbert space and $\mathcal{V} \subset \mathcal{H}$ be any subspace. In our analysis of the P-greedy algorithm's convergence we will use the notation

$$\Pi_V : \mathcal{H} \to \mathcal{V}$$

for a generalized orthogonal projection operator onto $\mathcal{V}$ and

$$P_{\mathcal{V}}(x) := \sup_{0 \neq f \in \mathcal{H}} \frac{|f(x) - \Pi_V(f)|}{\|f\|}$$

for a generalized Power Function. In the case that $\mathcal{H}$ is a reproducing kernel Hilbert space $\mathcal{H}_K(\Omega)$ an analogous characterization to lemma 3.4 follows immediately for any subspace $\mathcal{V} \subset \mathcal{H}_K(\Omega)$.

3.4. Greedy Algorithms

In general, the number N of available data sites $x_1, ..., x_N \in \Omega \subset \mathbb{R}^d$ can be very large. This might be a problem for an efficient computation of the entire expansion

$$s_f = \sum_{i=1}^{N} K(\cdot, x_i)\alpha_i$$

of all N data sites for a target model $f : \Omega \to \mathbb{R}^q$. Additionally, evaluating a large expansion would be very expensive in practice. Therefore, interpolating on only a subset of all available data sites and computing a sparse expansion is a commonly used strategy. Yet, finding an optimal selection of data sites is a combinatorially and computationally prohibitive task. Greedy techniques provide a valid alternative. With greedy algorithms, one finds a selection of $n << N$ data sites $x_{j_1}, ..., x_{j_n}$ such that the sparse surrogate model

$$f_n = \sum_{i=1}^{n} \alpha_i K(\cdot, x_{j_i})$$

is ideally as good as the full surrogate model s_f. The general procedure is as follows. Let $x_1, ..., x_N \in \Omega$ denote all available data sites. Start off with an empty selection of points $X_0 = \emptyset$ and hence indexes $J_0 = \emptyset$. In the n-th iteration choose a data value x_{j_n} by some selection criterion and add it to the previous selection

$$J_n = J_{n-1} \cup \{j_n\}, \quad X_n = X_{n-1} \cup \{x_{j_n}\}.$$

For each iteration compute the surrogate model f_n given by the interpolation conditions

$$f_n(x) = f(x) \quad \forall x \in X_n.$$

After each iteration the quality of the approximation is examined and once an error threshold is met, the algorithm stops.

Different selection criteria produce different interpolants. A few criteria are presented below. We will denote by $(z)_j$ the j-th component of $z \in \mathbb{R}^q$.

The *P-greedy algorithm* selects the point x_{j_n} in the n-th iteration that maximizes the power function $P_{V(X_{n-1})}$ of the previous iteration. To be precise

$$x_{j_n} := \operatorname*{arg\,max}_{x \in \Omega \setminus X_{n-1}} P_{V(X_{n-1})}(x)$$

for $n \geq 1$. This selection rule minimizes $\max_{1 \leq i \leq N} |P_{V(X_n)}(x_i)|$.

The *f-greedy algorithm* selects the point x_{j_n} for which the 2-norm of the residual $r_{n-1}(x) := f(x) - f_{n-1}(x)$ of the previous iteration reaches its maximum:

$$x_{j_n} := \operatorname*{arg\,max}_{x \in \Omega \setminus X_{n-1}} \sum_{i=1}^{q} |(r_{n-1}(x))_i|^2.$$

The *f/P-greedy algorithm* is an improved version of the *f-greedy algorithm* and it selects x_{j_n} that maximizes the ratio between the residual and the power function:

$$x_{j_n} := \operatorname*{arg\,max}_{x \in \Omega \setminus X_{n-1}} \frac{\sum_{i=1}^{q} |(r_{n-1}(x))_i|^2}{P_{V(X_{n-1})}(x)}.$$

Since the power function only vanishes on X_{n-1}, this selection is well defined.

Remark. In these selection criteria, we assume that the maxima will always be attained. For continuous f and K and compact Ω this is guaranteed. If the maximum is attained by more than one point, the selection is arbitrary. Thus, the point sets defined by these algorithms are not unique.

A notable difference between the P-greedy algorithm and the other two is that the P-greedy selection does not depend on the values of f. It selects the points by merely considering Ω and their values of K and P. This algorithm thus produces point sets X_n that are data-independent and can be used for approximating *any* model $f \in \mathcal{N}_K(\Omega)^q$, whereas the f- and f/P-greedy algorithms choose their point sets for a specific $f \in \mathcal{N}_K(\Omega)^q$.

In this thesis, we will concentrate on the P-greedy algorithm as a realization of kernel interpolation. We will further investigate the algorithm in the next section, examine its convergence in chapter 4 and see an implementation in the chapter 5.

3.5. The P-Greedy Algorithm

Let $x_1, ..., x_N \in \Omega \subset \mathbb{R}^d$ be N pairwise distinct data sites from Ω, fix a strictly positive definite kernel K on Ω and a model $f : \Omega \to \mathbb{R}^q$ such that $f \in \mathcal{N}_K(\Omega)^q$. In order to implement our studied kernel interpolation of f on $x_1, ..., x_N$, we will take a closer look at the data-independent P-greedy algorithm that we just introduced. In preparation for the implementation in chapter 5, we will prove some useful characterizations of the respective surrogate models, Power Function and residuals.

The P-greedy algorithm chooses point sets $X_n := \{x_{j_1}, ..., x_{j_n}\}$ from the available data sites $x_1, ..., x_N$ the following manner. In the first iteration, it chooses x_{j_1} such that

$$x_{j_1} := \operatorname*{arg\,max}_{x \in \Omega} P_{V(X_0)}(x).$$

Using characterization 3.4 for $X_0 = \emptyset$

$$P_{V(X_0)}(x) = \|K(\cdot, x)\|$$

and computing $\|K(\cdot, x)\|$ explicitly by applying the kernel's reproducing property, we obtain

$$x_{j_1} = \operatorname*{arg\,max}_{x \in \Omega} \sqrt{K(x, x)}.$$

For translational invariant kernels K, i.e. kernels such that

$$K(x, y) = \Phi(x - y)$$

for a function $\Phi : \mathbb{R}^d \to \mathbb{R}$, x_{j_1} can be chosen arbitrarily since $K(x, x) = \Phi(0)$ is constant for all $x \in \Omega$. Once $X_{n-1} = \{x_{j_1}, ..., x_{j_{n-1}}\}$ and $J_{n-1} = \{j_1, ..., j_{n-1}\}$ are chosen, it chooses x_{j_n} such that

$$x_{j_n} = \operatorname*{arg\,max}_{x \in \Omega \setminus X_{n-1}} P_{V(X_{n-1})}(x)$$

and set $J_n := J_{n-1} \cup \{j_n\}$ and $X_n := X_n \cup \{x_{j_n}\}$. Note that these point selections are in general not unique as the maximum of the Power Function is not.

To compute the kernel interpolants

$$f_n = \sum_{i=1}^{n} \alpha_i K(\cdot, x_{j_i})$$

for the point sets X_n the way we have described so far, we would have to solve the entire linear system

$$A_{K,X_n}\alpha = f$$

in each iteration. This would result in computing all n coefficients of α in each iteration. The previously computed $\alpha_1, ..., \alpha_{n-1}$ have to be recomputed in the n-the iteration. Furthermore, the bases $(K(\cdot, x_1), ..., K(\cdot, x_n))$ are unstable and the corresponding kernel matrices A_{K,X_n} ill-conditioned for $n \in \mathbb{N}$. The authors of [8] and [7] have analyzed different bases of the subspaces $V(X_n)$, $n \in \mathbb{N}$, and introduced a so called *Newton basis* as a valid alternative. The Newton bases are nested orthonormal bases $(v_1, ..., v_n)$ of the subspaces $V(X_n)$ for $n \in \mathbb{N}$. In our interpolation algorithm, we will construct these Newton bases in the following manner. Since each component $f_n^{(k)}$ is the orthogonal projection of each component $f^{(k)}$ into $V(X_n)$, we can write the interpolants as

$$f_n^{(k)}(x) = \sum_{i=1}^{n} c_i^{(k)} v_i(x), \quad k = 1, ..., q,\ n \in \mathbb{N}$$

for $(v_1, ..., v_n)$ the Newton basis of $V(X_n)$ and the coefficients

$$c_i^{(k)} = \langle f^{(k)}, v_i \rangle, \quad i = 1, ..., n,\ k = 1, ..., q.$$

Let's denote the vector of all coefficients by

$$c_i := \begin{pmatrix} c_i^{(1)} \\ \vdots \\ c_i^{(q)} \end{pmatrix} \in \mathbb{R}^q.$$

In the n-th iteration, the previously determined basis functions $v_1, ..., v_{n-1}$ and coefficients $c_1, ..., c_{n-1}$ can then be reused and solely v_n and c_n need to be computed. In the implementation we can thus use the recursive formulation

$$f_n = f_{n-1} + c_n v_n \text{ and } f_0 := 0 \tag{3.5}$$

for $n \in \mathbb{N}$.

For the Power Function we can deduce the following. By using the characterization in lemma 3.4, writing the interpolant in terms of this basis and applying the kernel's reproducing property we get

$$\begin{aligned} P_{V(X_n)}(x)^2 &= \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\|^2 \\ &= \langle K(\cdot, x) - \sum_{k=1}^{n} v_k(x) v_k, K(\cdot, x) - \sum_{l=1}^{n} v_l(x) v_l \rangle \\ &= K(x, x) - \sum_{k=1}^{n} v_k(x) \sum_{l=1}^{n} v_l(x) \langle v_k, v_l \rangle \end{aligned}$$

for $x \in \mathbb{R}^d$. Since our Newton basis is orthonormal, we obtain

$$P_{V(X_n)}(x)^2 = K(x,x) - \sum_{k=1}^{n} v_k(x)^2, \quad x \in \mathbb{R}^d. \tag{3.6}$$

This results in the following recursive formulation of the Power Function.

Corollary 3.7. *The Power Function satisfies*

$$P_{V(X_n)}(x)^2 = P_{V(X_{n-1})}(x)^2 - v_n(x)^2$$

for $x \in \Omega$ and any orthonormal basis $(v_1, v_2, ..., v_n)$ of $V(X_n)$. Thus,

$$\left\| P_{V(X_n)} \right\|_{L_\infty(\Omega)} \leq \left\| P_{V(X_{n-1})} \right\|_{L_\infty(\Omega)},$$

holds, i.e. the Power Function is decreasing in each iteration of the P-greedy algorithm.

Now, let's examine how the Newton basis is constructed. The nested orthonormal bases $(v_1, ..., v_n)$ of the subspaces $V(X_n)$ are obtained by applying the Gram-Schmidt orthonormalization procedure to the bases $(K(\cdot, x_{j_1}), ..., K(\cdot, x_{j_n}))$ for $n \in \mathbb{N}$. Hence, the first basis function $v_1 : \Omega \to \mathbb{R}$ is given by

$$v_1 := \frac{K(\cdot, x_{j_1})}{\|K(\cdot, x_{j_1})\|} = \frac{K(\cdot, x_{j_1})}{\sqrt{K(x_{j_1}, x_{j_1})}}.$$

Define

$$\begin{aligned} v_n' &:= K(\cdot, x_{j_n}) - \sum_{i=1}^{n-1} \langle K(\cdot, x_{j_n}), v_i \rangle v_i \\ &= K(\cdot, x_{j_n}) - \sum_{i=1}^{n-1} v_i(x_{j_n}) v_i, \end{aligned}$$

then the n-th basis function $v_n : \Omega \to \mathbb{R}$ is determined by

$$v_n = \frac{v_n'}{\|v_n'\|}.$$

To explicitly compute evaluations of the basis functions, we will take a closer look at $\|v_n'\|$.

Proposition 3.8. *The $\mathcal{N}_K(\Omega)$-norm of v_n' can be computed as*

$$\|v_n'\| = P_{V(X_{n-1})}(x_{j_n}).$$

Proof. Applying the kernel's reproducing property, bilinearity of the inner product and

orthonormality of the basis yields

$$\begin{aligned}
\|v_n'\|^2 &= \left\| K(\cdot, x_{j_n}) - \sum_{i=1}^{n-1} v_i(x_{j_n}) v_i \right\|^2 \\
&= \|K(\cdot, x_{j_n})\|^2 - 2\langle K(\cdot, x_{j_n}), \sum_{i=1}^{n-1} v_i(x_{j_n}) v_i \rangle + \langle \sum_{i=1}^{n-1} v_i(x_{j_n}) v_i, \sum_{j=1}^{n-1} v_j(x_{j_n}) v_j \rangle \\
&= K(x_{j_n}, x_{j_n}) - 2 \sum_{i=1}^{n-1} v_i(x_{j_n})^2 + \sum_{i=1}^{n-1} v_i(x_{j_n}) \sum_{j=1}^{n-1} v_j(x_{j_n}) \langle v_i, v_j \rangle \\
&= K(x_{j_n}, x_{j_n}) - \sum_{i=1}^{n-1} v_i(x_{j_n})^2.
\end{aligned}$$

By equation (3.6)

$$\|v_n'\| = P_{V(X_{n-1})}(x_{j_n})$$

follows. □

Therefore, we can implement the basis functions as

$$v_n = \frac{K(\cdot, x_{j_n}) - \sum_{i=1}^{n-1} v_i(x_{j_n}) v_i}{P_{V(X_{n-1})}(x_{j_n})}, \quad n \geq 2. \tag{3.7}$$

To estimate the quality of our interpolants, we define the residual in the n-th iteration by

$$r_n := f - f_n : \mathbb{R}^d \longrightarrow \mathbb{R}^q$$

where $r_0 := f$. Using the recursive formulation (3.5), we can deduce

$$r_n = r_{n-1} - c_n v_n \tag{3.8}$$

for $n \geq 1$.

Furthermore, we can prove a useful characterization of the interpolant's coefficients with respect to the Newton basis, i.e. the coefficients $c_i \in \mathbb{R}^q$, $i = 1, ..., n$, such that

$$f_n = \sum_{i=1}^{n} c_i v_i.$$

Proposition 3.9. *The n-th coefficient of the interpolant f_n with respect to the Newton basis can be computed as*

$$c_n = \frac{r_{n-1}(x_{j_n})}{P_{V(X_{n-1})}(x_{j_n})}.$$

Proof. Since $(v_1, ..., v_n)$ is orthonormal and f_n is the orthogonal projection of f onto $V(X_n)$, we can compute c_n as

$$c_n = \begin{pmatrix} \langle f^{(1)}, v_n \rangle \\ \vdots \\ \langle f^{(q)}, v_n \rangle \end{pmatrix}.$$

Hence, we obtain

$$\begin{aligned} c_n^{(k)} &= \langle f^{(k)}, \frac{v_n'}{\|v_n'\|} \rangle \\ &= \frac{1}{P_{V(X_{n-1})}(x_{j_n})} \langle f^{(k)}, K(\cdot, x_{j_n}) - \sum_{i=1}^{n-1} v_i(x_{j_n}) v_i \rangle \\ &= \frac{1}{P_{V(X_{n-1})}(x_{j_n})} (f^{(k)}(x_{j_n}) - \sum_{i=1}^{n-1} \langle f^{(k)}, v_i \rangle v_i(x_{j_n})) \\ &= \frac{f^{(k)}(x_{j_n}) - f_{n-1}^{(k)}(x_{j_n})}{P_{V(X_{n-1})}(x_{j_n})} \end{aligned}$$

by using proposition 3.8 and applying the kernel's reproducing property. □

Once the final number of selected points $x_{j_1}, ..., x_{j_{n_0}}$ is reached and the final interpolant

$$f_{n_0} = \sum_{i=1}^{n_0} c_i v_i$$

is computed, we would like to deduce the coefficients $\alpha_1, ..., \alpha_{n_0}$ with respect to the kernel basis $(K(\cdot, x_{j_1}), ..., K(\cdot, x_{j_{n_0}}))$:

$$f_{n_0} = \sum_{i=1}^{n_0} \alpha_i K(\cdot, x_{j_i}).$$

In order to compute those, we need to compute the transition matrix between our two bases.

Proposition 3.10. *The j-th column of the transition matrix $A = (a_{ij})_{i,j=1,...,n_0}$ from $(v_1, ..., v_{n_0})$ to $(K(\cdot, x_{j_1}), ..., K(\cdot, x_{j_{n_0}}))$ can be computed as*

$$\begin{aligned} a_{ij} &= -\frac{1}{P_{V(X_{j-1})}(x_{j_j})} \sum_{k=1}^{j-1} a_{ik} v_k(x_{j_j}), && i = 1, ..., j-1, \\ a_{ij} &= \frac{1}{P_{V(X_{j-1})}(x_{j_j})}, && i = j \\ a_{ij} &= 0, && i = j+1, ..., n_0. \end{aligned}$$

The coefficients $\alpha_1, ..., \alpha_{n_0} \in \mathbb{R}^q$ with respect to the Newton basis can be computed as

$$\alpha_i^{(k)} = \sum_{j=1}^{n_0} a_{ij} c_j^{(k)} \quad \forall i = 1, ..., n_0 \; k = 1, ..., q.$$

Proof. The transition matrix is the unique matrix $A = (a_{ij})_{i,j=1,...,n_0}$ that satisfies

$$v_j = \sum_{i=1}^{n_0} a_{ij} K(\cdot, x_{j_i}).$$

Since

$$v_1 = \frac{K(\cdot, x_{j_1})}{\sqrt{K(x_{j_1}, x_{j_1})}} = \frac{1}{P_{V(X_0)}(x_{j_1})} K(\cdot, x_{j_1}),$$

it follows that

$$a_{11} = \frac{1}{P_{V(X_0)}(x_{j_1})} \quad \text{and} \quad a_{i1} = 0, \quad i = 2, ..., n_0.$$

Now consider $j > 1$. Assume we have already shown the characterization for columns $k = 1, ..., j-1$. By equation (3.7) we have

$$\begin{aligned} v_j &= \frac{K(\cdot, x_{j_j}) - \sum_{k=1}^{j-1} v_k(x_{j_j}) v_k}{P_{V(X_{j-1})}(x_{j_j})} \\ &= \frac{1}{P_{V(X_{j-1})}(x_{j_j})} K(\cdot, x_{j_j}) - \sum_{k=1}^{j-1} \frac{v_k(x_{j_j})}{P_{V(X_{j-1})}(x_{j_j})} v_k. \end{aligned}$$

Since A is the transformation matrix,

$$v_k = \sum_{l=1}^{n} a_{lk} K(\cdot, x_{j_l}) \quad \text{for } k = 1, ..., j-1$$

holds and

$$v_j = \frac{1}{P_{V(X_{j-1})}(x_{j_j})} K(\cdot, x_{j_j}) - \sum_{l=1}^{n} \frac{1}{P_{V(X_{j-1})}(x_{j_j})} \sum_{k=1}^{j-1} a_{lk} v_k(x_{j_j}) K(\cdot, x_{j_l})$$

follows. Since we assume that the characterization holds for the columns $k = 1, ..., j-1$,

$$a_{lk} = 0 \quad \text{for } l = j, ..., n_0.$$

Hence,

$$v_j = \frac{1}{P_{V(X_{j-1})}(x_{j_j})} K(\cdot, x_{j_j}) - \sum_{l=1}^{j-1} \frac{1}{P_{V(X_{j-1})}(x_{j_j})} \sum_{k=1}^{j-1} a_{lk} v_k(x_{j_j}) K(\cdot, x_{j_l})$$

and by equating the coefficients the claim follows.

For the coefficients $\alpha_1, ..., \alpha_{n_0}$ hence follows

$$f_{n_0}^{(k)} = \sum_{j=1}^{n_0} c_j^{(k)} v_j = \sum_{i=1}^{n_0} \sum_{j=1}^{n_0} a_{ij} c_j^{(k)} K(\cdot, x_i)$$

and by equating the coefficients

$$\alpha_i^{(k)} = \sum_{j=1}^{n_0} a_{ij} c_j^{(k)} \quad \text{for } i = 1, ..., n_0.$$

□

Furthermore, recall from equation 3.2 that we have

$$\left|f^{(k)}(x) - f_n^{(k)}(x)\right| \leq P_{V(X_n)}(x) \left\|f^{(k)}\right\|_{\mathcal{N}_K(\Omega)}$$

and hence

$$\|r_n(x)\|_2 = \|f(x) - f_n(x)\|_2 \leq P_{V(X_n)}(x) \|f\|_{\mathcal{N}_K(\Omega)^q}. \tag{3.9}$$

for every $x \in \mathbb{R}^d$. Therefore, $\left\|P_{V(X_n)}\right\|_\infty \|f\|_{\mathcal{N}_K(\Omega)^q}$ is an upper bound for the residual in every iteration n.

In our last proposition, we will prove how to compute the $\mathcal{N}_K(\Omega)^q$-norm of the target function $f \in \mathcal{N}_K(\Omega)^q$. For notational reasons, we will first look at the scalar-valued case.

Proposition 3.11. *Let $f \in \mathcal{N}_K(\Omega)$. Then,*

$$\|f - f_n\|^2_{\mathcal{N}_K(\Omega)} = \|f\|^2_{\mathcal{N}_K(\Omega)} - \|f_n\|^2_{\mathcal{N}_K(\Omega)}.$$

Proof. Since $f_n = \Pi_{V(X_n)}(f)$, we have

$$\begin{aligned}\|f - f_n\|^2 &= \langle f - \Pi_{V(X_n)}(f), f - \Pi_{V(X_n)}(f)\rangle \\ &= \|f\|^2 - \langle f, \Pi_{V(X_n)}(f)\rangle - \langle f - \Pi_{V(X_n)}(f), \Pi_{V(X_n)}(f)\rangle\end{aligned}$$

and since $f - \Pi_{V(X_n)}(f) \in V(X_n)^\perp$ and $\Pi_{V(X_n)}(f) = \sum_{i=1}^n \langle f, v_i\rangle v_i$ for the orthonormal Newton basis $(v_1, ..., v_n)$ of $V(X_n)$, it follows that

$$\begin{aligned}\|f - f_n\|^2 &= \|f\|^2 - \langle f, \Pi_{V(X_n)}(f)\rangle \\ &= \|f\|^2 - \sum_{i=1}^n \langle f, v_i\rangle^2 \\ &= \|f\|^2 - \left\|\Pi_{V(X_n)}(f)\right\|^2.\end{aligned}$$

□

Thus, for $f \in \mathcal{N}_K(\Omega)^q$ we have

$$\|f - f_n\|^2_{\mathcal{N}_K(\Omega)^q} = \sum_{k=1}^q \left\|f^{(k)} - f_n^{(k)}\right\|^2_{\mathcal{N}_K(\Omega)}.$$

And hence

$$\|f - f_n\|^2_{\mathcal{N}_K(\Omega)^q} = \sum_{k=1}^q \left(\left\|f^{(k)}\right\|^2_{\mathcal{N}_K(\Omega)} - \left\|f_n^{(k)}\right\|^2_{\mathcal{N}_K(\Omega)}\right) = \|f\|^2_{\mathcal{N}_K(\Omega)^q} - \|f_n\|^2_{\mathcal{N}_K(\Omega)^q}. \tag{3.10}$$

We now have all tools and equations at hand to implement the P-greedy algorithm in chapter 5. However to justify the implementation, we will first take a look at the theoretical qualities of the P-greedy algorithm in the next chapter and prove that it converges at the rates of the best known (non-greedy) approaches.

CHAPTER 4

Convergence Result for the P-Greedy Algorithm

In the last chapter, we have introduced and investigated the P-greedy algorithm as a realization of kernel interpolation. To justify its implementation in the next chapter, we will examine it closer from a mathematical point of view and show its convergence.

We have introduced the Power Function $P_{V(X_n)}$ in 3.3. As it measures the algorithm's interpolation quality, we will examine its convergence. We will see in this chapter that through the P-greedy algorithm, the Power Function decays as good as for any other selection algorithm. For this convergence analysis, we will follow the work of [10].

4.1. Convergence Rates for the Best Known (Non-Greedy) Point Selection

In [11] the decay of the Power Function was studied and related to the kernel's smoothness and the point distribution of the point selection in Ω. In this section, we will show what we precisely mean by that and in the next section that the P-greedy algorithm meets this decay rate.

Assume that Ω satisfies an interior cone condition and that K is a translational invariant kernel, i.e. such that $K(x,y) = \Phi(x-y)$ for a function $\Phi : \mathbb{R} \to \mathbb{R}$ that has a continuous Fourier transform $\widehat{\Phi}$.

Definition 4.1. *Let $X_n \subset \Omega$ be any set of n pairwise distinct points in Ω. The fill distance of X_n is defined as*

$$h_{X_n,\Omega} := \sup_{x\in\Omega} \min_{y\in X_n} \|x-y\|_2 ,$$

where $\|\cdot\|_2$ denotes the Euclidean norm in $\mathbb{R}^d$.

In [11] two cases were taken into account. In the first, kernels of *finite smoothness* β were considered. By this we will mean kernels such that

$$\widehat{\Phi} \sim (1+\|w\|_2^2)^{-\beta},$$

i.e. kernels for which there exist constants $c_1, c_2 > 0$ and $\beta \in \mathbb{N},\ \beta > \frac{d}{2}$ such that

$$c_1(1+\|w\|_2^2)^{-\beta} \le \widehat{\Phi}(w) \le c_2(1+\|w\|_2^2)^{-\beta}.$$

In this case, $K \in \mathcal{C}^\beta$ and $\mathcal{N}_K(\Omega)$ is norm equivalent to the Sobolev space $H^\beta(\mathbb{R}^d)$, see corollary 2.18. This concerns for example Wendland kernels. In the second case, kernels of infinite smoothness were considered, for example Gaussian kernels. For these two cases the following decay rates of the Power Function were shown.

Theorem 4.2. *Let K and Ω be as assumed above and X_n any set of n pairwise distinct points in Ω. There exists constants $\hat{c_1}, \hat{c_2}, \hat{c_3} \in \mathbb{R}$ independent of X_n such that*

(1) For K of finite smoothness $\beta \in \mathbb{N}$:

$$\left\| P_{V(X_n)} \right\|_{L_\infty(\Omega)} \leq \hat{c_1} h_{X_n,\Omega}^{\beta - d/2}.$$

(2) For K of infinite smoothness:

$$\left\| P_{V(X_n)} \right\|_{L_\infty(\Omega)} \leq \hat{c_2} e^{-\hat{c_3}/h_{X_n,\Omega}}.$$

In particular, this holds for point sets X_n that are uniformly distributed in Ω. For those

$$h_{X_n,\Omega} \leq c n^{-1/d}$$

holds for a constant $c \in \mathbb{R}$, $c > 0$ independent of n. In this case we can write the decay rate in terms of n.

Corollary 4.3. *Let K and Ω be as assumed above. There exist point sets X_n of n pairwise distinct points in Ω and constants $c_1, c_2, c_3 \in \mathbb{R}$ independent of n such that*

(1) For K of finite smoothness $\beta \in \mathbb{N}$:

$$\left\| P_{V(X_n)} \right\|_{L_\infty(\Omega)} \leq c_1 n^{-\frac{\beta}{d} + \frac{1}{2}}.$$

(2) For K of infinite smoothness:

$$\left\| P_{V(X_n)} \right\|_{L_\infty(\Omega)} \leq c_2 e^{-c_3 n^{1/d}}.$$

These decay rates set the benchmark for our convergence analysis of the P-greedy algorithm.

4.2. Convergence Analysis of the P-Greedy Algorithm

The authors of [5] have studied the convergence of the P-greedy algorithm before and came to the following result.

Theorem 4.4. *Let $\Omega \subset \mathbb{R}^d$ be compact and satisfying an interior cone condition, $\Omega_1 \supset \Omega$ compact and convex and $K \in \mathcal{C}^2$ a twice differential strictly positive definite kernel. Then the P-greedy algorithm produces point sets X_n such that*

$$\left\| P_{V(X_n)} \right\|_{L\infty(\Omega)} \leq c n^{-\frac{1}{d}}$$

for all $n \in \mathbb{N}$ and a suitable constant $c \in \mathbb{R}$ independent of n.

We will find similar estimates in this section, yet are able to improve the bounds for kernels of greater smoothness.

We will now have a look at the Kolmogorov width of the subset $\mathcal{V}$ of all possible kernel translates in $\mathcal{H}_K(\Omega)$ and draw the connection to the Power Function. A result from [1] will yield the convergence rate of the P-greedy algorithm.

Here as well, we will consider translational invariant kernels.

4.2.1. The Kolmogorov n-Width and its Connection to the Power Function

Definition 4.5. *For a Hilbert space $\mathcal{H}$, the Kolmogorov n-width of a subset $\mathcal{V} \subset \mathcal{H}$ is defined as*

$$d_n(\mathcal{V}, \mathcal{H}) := \inf_{\substack{V \subset \mathcal{H}, \\ dimV = n}} \sup_{f \in \mathcal{V}} \|f - \Pi_V(f)\| .$$

For subsets $\mathcal{V}, \mathcal{W} \subset \mathcal{H}$ and $f \in \mathcal{V}$, we will denote by

$$E(f, \mathcal{W}) := \inf_{g \in \mathcal{W}} \|f - g\| = \|f - \Pi_{\mathcal{W}}(f)\|$$

the approximation error of an element $f \in \mathcal{V}$ by elements in $\mathcal{W}$ and by

$$E(\mathcal{V}, \mathcal{W}) := \sup_{f \in \mathcal{V}} E(f, \mathcal{W}) = \sup_{f \in \mathcal{V}} \|f - \Pi_{\mathcal{W}}(f)\|$$

the maximum approximation error of elements in $\mathcal{V}$ by elements in $\mathcal{W}$. Hence, we can understand the Kolmogorv n-width

$$d_n(\mathcal{V}, \mathcal{H}) = \inf_{\substack{V \subset \mathcal{H}, \\ \dim V = n}} E(\mathcal{V}, V)$$

as the smallest worst case approximation error of an element in $\mathcal{V}$ by any n-dimensional subspace of $\mathcal{H}$. If there is no ambiguity, we will denote the Kolmogorv n-width simply by d_n.

A thorough study of the Kolmogorov n-width can be found in [9]. At this point, we will only need the following property.

Lemma 4.6. *Let $\mathcal{H}$ be a Hilbert space and $A \subset \mathcal{H}$ a closed subset. A is compact in $\mathcal{H}$ if and only if A is bounded and* $\lim_{n \to \infty} d_n(A, \mathcal{H}) = 0$.

Proof. Let A be compact and $\epsilon > 0$. There exist $n_0 \in \mathbb{N}$ and $a_1, ..., a_{n_0}$ pairwise distinct such that

$$A \subset \bigcup_{i=1}^{n_0} B_\epsilon(a_i),$$

where $B_\epsilon(a_i)$ denotes the open ball around a_i with radius ϵ. Then, we have

$$\|a\| \leq \|a - a_i\| + \|a_i\| \leq \epsilon + \max_{1 \leq i \leq n_0} \|a_i\| < \infty$$

for all $a \in A$. Hence, A is bounded. Define the n_0-dimensional subspace $A_{n_0} := \text{span}\{a_1, ..., a_{n_0}\}$. Then

$$d_{n_0} = \inf_{\substack{V \subset \mathcal{H}, \\ \dim V = n_0}} E(A, V) \leq E(A, A_{n_0}) < \epsilon.$$

Since for all $n \geq n_0$ we have

$$0 \leq d_n \leq d_{n_0} < \epsilon,$$

the first implication is proven.

Now let A be bounded and $\lim_{n\to\infty} d_n = 0$. Let $\epsilon > 0$. There exists $n_0 \in \mathbb{N}$ such that $d_n \leq \frac{\epsilon}{2} =: \epsilon'$ for all $n \geq n_0$. Fix $n \geq n_0$. Since

$$\epsilon' > d_n = \inf_{\substack{V \subset \mathcal{H}, \\ \dim V = n}} E(A, V),$$

there exists an n-dimensional subspace A_n such that $E(A, A_n) < \epsilon'$, i.e. for all $x \in A$ there exists an $y \in A_n$ such that $\|x - y\| < \epsilon'$ and thus

$$\|y\| = \|x - (x - y)\| < d_0 + \epsilon'.$$

Since A is bounded,

$$d_0 = \sup_{x \in A} \|x\| < \infty$$

and the set

$$Y := \{y \in A_n | \, \|y\| \leq d_0 + \epsilon'\}$$

is a closed and bounded subset of the finite dimensional vector space A_n, hence compact. So, there exist $y_1, ..., y_k \in Y$ such that

$$Y \subset \bigcup_{i=1}^{k} B_\epsilon(y_i).$$

Now let $x \in A$. Then there exists $y \in Y$ such that $\|x - y\| < \epsilon'$. Yet as Y has a finite subcover, there exists $y_i, i \in \{1, ..., k\}$ such that $\|y - y_i\| \leq \epsilon'$. We get

$$\|x - y_i\| \leq \|x - y\| + \|y - y_i\| < 2\epsilon' = \epsilon.$$

So, $(B_\epsilon(y_i))$ for $y_1, ..., y_k \in A_n$ is a finite subcover for A.

□

In our setting, consider a reproducing kernel Hilbert space $\mathcal{H}_K(\Omega)$. For any subset $\tilde{\Omega} \subset \Omega$ let $\mathcal{V}(\tilde{\Omega})$ be defined as

$$\mathcal{V}(\tilde{\Omega}) := \{K(\cdot, x) \mid x \in \tilde{\Omega}\} \subset \mathcal{H}_K(\Omega).$$

We deduce the following connection between the Kolmogorov n-width and the Power Function.

Lemma 4.7. *Let $\tilde{\Omega} \subset \Omega$ be a closed subset. If there exists a sequence of point sets $X_n \subset \Omega$ and a sequence $(\gamma_n)_n \subset \mathbb{R}$ such that for all $n \in \mathbb{N}$*

$$\left\| P_{V(X_n)} \right\|_{L_\infty(\tilde{\Omega})} \leq \gamma_n,$$

then

$$d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) \leq \gamma_n.$$

holds.

Moreover, if $\lim_{n\to\infty} \gamma_n = 0$, then $\mathcal{V}(\tilde{\Omega})$ is compact in $\mathcal{H}_K(\Omega)$.

Proof. Using the characterization of the generalized Power Function in lemma 3.4, we get for any n-dimensional subspace V of $\mathcal{H}_K(\Omega)$

$$\begin{aligned} E(\mathcal{V}(\tilde{\Omega}), V) &= \sup_{f \in V(\tilde{\Omega})} \|f - \Pi_V(f)\| = \sup_{x \in \tilde{\Omega}} \|K(\cdot, x) - \Pi_V(K(\cdot, x))\| \\ &= \sup_{x \in \tilde{\Omega}} P_V(x) = \|P_V\|_{L_\infty(\tilde{\Omega})}. \end{aligned}$$

Taking the infimum over only n-dimensional subset of the form $V(X_n) = \text{span}\{K(\cdot, x_1), ..., K(\cdot, x_n)\}$, where $|X_n| = n$, and taking the L_∞-norm over the super set Ω yield the following inequality

$$\begin{aligned} d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) &= \inf_{\substack{V \subset \mathcal{H}_K(\Omega), \\ \dim V = n}} E(\mathcal{V}(\tilde{\Omega}), V) \\ &= \inf_{\substack{V \subset \mathcal{H}_K(\Omega), \\ \dim V = n}} \|P_V\|_{L_\infty(\tilde{\Omega})} \\ &\leq \inf_{X_n \subset \Omega} \left\|P_{V(X_n)}\right\|_{L_\infty(\tilde{\Omega})} \\ &\leq \inf_{X_n \subset \tilde{\Omega}} \left\|P_{V(X_n)}\right\|_{L_\infty(\Omega)}. \end{aligned}$$

for all $n \in \mathbb{N}$. If there exists a sequence of point sets $X_n \subset \Omega$ and a sequence $(\gamma_n)_n \subset \mathbb{R}$ such that for all $n \in \mathbb{N}$

$$\left\|P_{V(X_n)}\right\|_{L_\infty(\Omega)} \leq \gamma_n,$$

we get

$$d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) \leq \left\|P_{V(X_n)}\right\|_{L_\infty(\Omega)} \leq \gamma_n.$$

In particular, if $\lim_{n\to\infty} \gamma_n = 0$, then also $d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega))$ converges to zero. Furthermore, $\mathcal{V}(\tilde{\Omega})$ is bounded: By applying the reproducing kernel property we get

$$\sup_{f \in \mathcal{V}(\Omega)} \|f\| = \sup_{x \in \tilde{\Omega}} \|K(\cdot, x)\| = \sup_{x \in \tilde{\Omega}} \sqrt{K(x, x)}$$

and since we assumed K to be translational invariant, there exists a function $\Phi : \mathbb{R} \to \mathbb{R}$ such that $K(x, y) = \Phi(x - y)$ which yields

$$\sup_{f \in \mathcal{V}(\Omega)} \|f\| = \sqrt{\Phi(0)} < \infty.$$

Moreover, since $\tilde{\Omega}$ is closed and K is continuous, also $\mathcal{V}(\tilde{\Omega})$ is closed. By lemma 4.6 it follows that $\mathcal{V}(\tilde{\Omega})$ is compact in $\mathcal{H}_K(\Omega)$. □

Applying this result to corollary 4.3 we get the following.

Corollary 4.8. *Let K and Ω be as assumed in section 4.1 and $\tilde{\Omega} \subset \Omega$ a closed subset. Then*

(1) For K of finite smoothness $\beta \in \mathbb{N}$:

$$d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) \leq c_1 n^{-\frac{\beta}{d} + \frac{1}{2}}.$$

(2) For K of infinite smoothness:

$$d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) \leq c_2 e^{-c_3 n^{1/d}}.$$

Furthermore, $\mathcal{V}(\tilde{\Omega})$ is compact in $\mathcal{H}_K(\Omega)$ in both cases.

Remark. Obviously the previous result hold for the case $\tilde{\Omega} = \Omega$. However, when implementing the algorithm one often has only access to a discretization of Ω or uses a restricted subset of Ω for performance reasons. Since we can explicitly give sharper bounds for a restricted set $\tilde{\Omega}$, we keep considering this case.

At this point, we need a general result for greedy algorithms in Hilbert spaces to continue our convergence analysis and deduce the algorithm's decay rate.

4.2.2. Convergence Analysis for Greedy Algorithms in Hilbert Spaces

We will now prove some results for the following setting. Let $\mathcal{H}$ be a separable Hilbert space, $\mathcal{F} \subset \mathcal{H}$ any compact subset and $f_1, f_2, ... \in \mathcal{F}$ a sequence of elements in $\mathcal{F}$ obtained by a greedy algorithm of the following form: Initially choose f_1 as

$$f_1 := \arg\max_{f \in \mathcal{F}} \|f\| .$$

Once $f_1, ..., f_n$ are chosen, consider $V_n := \text{span}\{f_1, ..., f_n\}$ and choose f_{n+1} as

$$f_{n+1} := \arg\max_{f \in \mathcal{F}} E(f, V_n).$$

Let $f_1^*, f_2^*, ..., f_n^*$ be the orthonormal basis of V_n obtained by applying Gram-Schmidt orthonormalization to $f_1, f_2, ..., f_n$. Note that for the orthogonal projection

$$\Pi_{V_n}(f) = \sum_{i=1}^{n} \langle f, f_i^* \rangle f_i^*$$

of $\mathcal{H}$ into V_n we have

$$f_n = \Pi_{V_n}(f_n).$$

Let σ_n denote the worst case approximation error for this greedy algorithm:

$$\sigma_n := E(\mathcal{F}, V_n) = \max_{f \in \mathcal{F}} E(f, V_n) = \max_{f \in \mathcal{F}} \|f - \Pi_{V_n}(f)\| .$$

Since $V_n \subset V_{n+1}$ for all $n \in \mathbb{N}$, the sequence $(\sigma_n)_n$ is monotonically decreasing. Obviously

$$d_n := d_n(\mathcal{F}, \mathcal{H}) \leq \sigma_n$$

follows since

$$d_n = \inf_{\substack{V \subset \mathcal{H}, \\ \dim V = n}} E(\mathcal{F}, V) \leq E(\mathcal{F}, V_n) = \sigma_n.$$

In what follows we will prove the following estimates of this algorithm's decay.

Theorem 4.9. *Let $\mathcal{H}$ be a Hilbert space and $\mathcal{F}$ one of its compact subsets. The following implications hold.*

(a) *If $d_n(\mathcal{F},\mathcal{H}) \leq C_0 n^{-\alpha}$ for all $n \in \mathbb{N}$, then*

$$\sigma_n \leq C_1 n^{-\alpha} \quad \forall n \in \mathbb{N}$$

with $C_1 := 2^{5\alpha+1} C_0$.

(b) *If $d_n(\mathcal{F},\mathcal{H}) \leq C_0 e^{-c_0 n^{\alpha}}$ for all $n \in \mathbb{N}$, then*

$$\sigma_n \leq C_2 e^{-c_1 n^{\alpha}} \quad \forall n \in \mathbb{N}$$

with $C_2 := \sqrt{2C_0}, c_1 := 2^{-1-2\alpha} c_0$.

For the proof of these estimates we will need a few technical lemmas.

Lemma 4.10. *Let $G = (g_{i,j})_{i,j=1,\dots,K}$ be a lower triangular $K \times K$ matrix with rows $g_1, \dots, g_K$ and W be any m-dimensional subspace of $\mathbb{R}^K$. Then the following inequality holds:*

$$\prod_{i=1}^{K} g_{i,i}^2 \leq \left(\frac{1}{m} \sum_{i=1}^{K} \|\Pi_W g_i\|_2^2 \right)^m \left(\frac{1}{K-m} \sum_{i=1}^{K} \|g_i - \Pi_W g_i\|_2^2 \right)^{K-m}.$$

Proof. Let $\phi_1, \dots, \phi_m$ be any orthonormal basis of W and $\phi_1, \dots, \phi_m, \phi_{m+1}, \dots, \phi_K$ its completion to an orthonormal basis of $\mathbb{R}^K$. Define the matrix $\Phi := (\phi_1 \dots \phi_K) \in \mathbb{R}^{K \times K}$. Consider the matrix $C := (c_{i,j})_{i,j=1,\dots,K}$ with columns $c_1, \dots, c_K$ defined as $C := G\Phi$. Its coefficients satisfy

$$c_{i,j} = \langle g_i, \phi_j \rangle_2.$$

Applying the inequality of arithmetic and geometric means to $\|c_1\|_2^2, \dots, \|c_m\|_2^2$ yields

$$\prod_{i=1}^{m} \|c_i\|_2^2 \leq \left(\frac{1}{m} \sum_{i=1}^{m} \|c_i\|_2^2 \right)^m = \left(\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{K} \langle g_j, \phi_i \rangle_2^2 \right)^m = \left(\frac{1}{m} \sum_{j=1}^{K} \|\Pi_W g_j\|_2^2 \right)^m \tag{4.1}$$

since $\phi_1, \dots, \phi_m$ is an orthonormal basis of W. Similarly we get

$$\prod_{i=m+1}^{K} \|c_i\|_2^2 \leq \left(\frac{1}{K-m} \sum_{i=m+1}^{K} \|c_i\|_2^2 \right)^{K-m} = \left(\frac{1}{K-m} \sum_{j=1}^{K} \|g_j - \Pi_W g_j\|_2^2 \right)^{K-m}. \tag{4.2}$$

Using Hadamard's inequality on C,

$$\det(C)^2 \leq \prod_{i=1}^{K} \|c_i\|_2^2,$$

and estimates (4.1) and (4.2) we obtain

$$\det(C)^2 \leq \left(\frac{1}{m} \sum_{j=1}^{K} \|\Pi_W g_j\|_2^2 \right)^m \left(\frac{1}{K-m} \sum_{j=1}^{K} \|g_j - \Pi_W g_j\|_2^2 \right)^{K-m}.$$

Yet

$$\det(C)^2 = \det(G)^2 \det(\Phi)^2 = \left(\prod_{i=1}^{K} g_{i,i}\right)^2$$

since $\phi_1, ..., \phi_K$ are orthonormal and G is a lower triangular matrix. We thus have proven the claim. □

Every infinite dimensional separable Hilbert space is isomorphic to the space ℓ_2 of squared summable sequences: Every separable Hilbert space $\mathcal{H}$ admits a countable orthonormal basis $(u_i)_{i\in\mathbb{N}}$. So one can form an isomorphism between $\mathcal{H}$ and ℓ_2 by extending the bijection between $(u_i)_{i\in\mathbb{N}}$ and the standard basis $(e_i)_{i\in\mathbb{N}}$ of ℓ_2, where $(e_i)_j = \delta_{ij}$. Thus, without loss of generality, we assume $\mathcal{H}$ to be ℓ_2 and $f_1^*, f_2^*, ...$ to be $e_1, e_2....$ For notational convenience, further assume $\mathcal{F}$ to be such that $\|f\| \leq 1$ for all $f \in \mathcal{F}$.

Consider the following matrix representation of these greedy algorithms. Let $A := (a_{ij})_{i,j=1,...,n}$ be defined as

$$a_{i,j} := \langle f_i, e_j \rangle$$

for $i, j \in \{1, ..., n\}, i \geq j$ and

$$a_{ij} := 0$$

for $j > i$. Then

$$f_i = \Pi_{V_i}(f_i) = \sum_{j=1}^{i} a_{i,j} e_j = \begin{pmatrix} a_{i,1} \\ \vdots \\ a_{i,i} \\ 0 \\ \vdots \end{pmatrix} \tag{4.3}$$

is represented in the i-th row of A. Note further that A is a lower triangular matrix for which we have

$$\begin{aligned} |a_{n,n}| &= |\langle f_n, e_n \rangle| = \|\langle f_n, e_n \rangle e_n\| \\ &= \left\| \sum_{i=1}^{n} \langle f_n, e_i \rangle e_i - \sum_{i=1}^{n-1} \langle f_n, e_i \rangle e_i \right\| \\ &= \|f_n - \Pi_{V_{n-1}} f_n\| = E(f_n, V_{n-1}) \end{aligned}$$

which lead by choice of f_n to the equation

$$|a_{n,n}| = E(f_n, V_{n-1}) = \max_{f\in\mathcal{F}} E(f, V_{n-1}) = \sigma_{n-1}. \tag{4.4}$$

And moreover,

$$\begin{aligned} \sum_{j=n}^{m} a_{m,j}^2 &= \langle f_m, e_m \rangle^2 + ... + \langle f_m, e_n \rangle^2 \\ &= \left\| \sum_{i=1}^{m} \langle f_m, e_i \rangle e_i - \sum_{i=1}^{n-1} \langle f_m, e_i \rangle e_i \right\|^2 \\ &= \|f_m - \Pi_{V_{n-1}} f_m\|^2 \end{aligned}$$

and hence

$$\sum_{j=n}^{m} a_{m,j}^2 \leq \sigma_{n-1}^2. \tag{4.5}$$

The next theorem and its deductions will provide our tools for the proof of theorem 4.9.

Theorem 4.11. *For any $N \geq 0,\ K \geq 1,\ 1 \leq m < K$, we have*

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

Proof. Let $N \geq 0,\ K \geq 1,\ 1 \leq m < K$. Consider the following $K \times K$ submatrix of A. Let $G = (g_{i,j})_{i,j=1,...,K}$ be defined as

$$G := \begin{pmatrix} g_1 \\ \vdots \\ g_K \end{pmatrix} := \begin{pmatrix} a_{N+1,N+1} & & 0 \\ \vdots & \ddots & \\ a_{N+K,N+1} & \dots & a_{N+K,N+K} \end{pmatrix}.$$

Every row g_i of G is thus the restriction of f_{N+i} to the coordinates $N+1, ..., N+i$.

Let $\mathcal{H}_m$ be the m-dimensional subspace of $\mathcal{H} = \ell_2$ such that

$$d_m = E(\mathcal{F}, \mathcal{H}_m).$$

For this $\mathcal{H}_m$ we have

$$E(f_{N+i}, \mathcal{H}_m) \leq \sup_{f \in \mathcal{F}} E(f, \mathcal{H}_m) = d_m.$$

Let $\tilde{W} \subset \ell_2$ be the restriction of $\mathcal{H}_m$ to $\operatorname{span}\{e_{N+1}, ..., e_{N+K}\}$, i.e.

$$\tilde{W} = \mathcal{H}_m \cap \operatorname{span}\{e_{N+1}, ..., e_{N+K}\}.$$

In general, $\tilde{W}$ has dimension $\leq m$. Let W be an m-dimensional subspace of $\operatorname{span}\{e_{N+1}, ..., e_{N+K}\}$ that contains $\tilde{W}$.

Equation (4.5) yields

$$\|\Pi_W g_i\|_{\ell_2}^2 \leq \|g_i\|_{\ell_2}^2 = \sum_{j=N+1}^{N+i} |a_{N+i,j}|^2 \leq \sigma_N^2. \tag{4.6}$$

And analogously

$$\|g_i - \Pi_W g_i\|_{\ell_2} \leq \|g_i - \Pi_{\tilde{W}} g_i\|_{\ell_2} = E(g_i, \tilde{W}) \leq E(f_{N+i}, \mathcal{H}_m) \leq d_m \tag{4.7}$$

since $\operatorname{span}\{e_1, ..., e_N, 0, ..., 0, e_{N+K+1}, ...\} \subset W^\perp \subset \tilde{W}^\perp$. After applying equation (4.4)

$$\prod_{i=1}^{K} \sigma_{N+i}^2 = \prod_{i=1}^{K} |a_{N+i,N+i}|^2 = \prod_{i=1}^{K} g_{i,i}^2,$$

we apply lemma 4.10 to our matrix G and subspace W and get

$$\prod_{i=1}^{K} \sigma_{N+i}^2 \leq \left(\frac{1}{m} \sum_{i=1}^{K} \|\Pi_W g_i\|_{\ell_2}^2\right)^m \left(\frac{1}{K-m} \sum_{i=1}^{K} \|g_i - \Pi_W g_i\|_{\ell_2}^2\right)^{K-m}.$$

Applying estimates (4.6) and (4.7) proves the claim:

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

□

Corollary 4.12. *Let $\mathcal{H}$ be a separable Hilbert space and $\mathcal{F}$ any of its compact subsets. For $n \geq 1$, we have*

$$\sigma_n \leq \sqrt{2} \min_{1\leq m<n} d_m^{\frac{n-m}{n}}$$

and in particular,

$$\sigma_{2n} \leq \sqrt{2d_n}.$$

Proof. Applying theorem 4.11 to $N = 0, K = n$ and any $1 \leq m < n$, we get

$$\prod_{i=1}^{n} \sigma_i^2 \leq \left(\frac{n}{m}\right)^m \left(\frac{n}{n-m}\right)^{n-m} \sigma_0^{2m} d_m^{2n-2m}.$$

Since σ_n is monotonically decreasing and $\sigma_0 = \max_{f\in\mathcal{F}} \|f\| \leq 1$ by assumption, we get

$$\sigma_n^{2n} \leq \left(\frac{n}{m}\right)^m \left(\frac{n}{n-m}\right)^{n-m} d_m^{2n-2m}.$$

It follows

$$\sigma_n \leq \left(\frac{n}{m}\right)^{\frac{m}{2n}} \left(\frac{n}{n-m}\right)^{\frac{n-m}{2n}} d_m^{\frac{n-m}{n}}.$$

Note that $\left(\frac{1}{x}\right)^x \left(\frac{1}{1-x}^{1-x} = x^{-x}(1-x)\right)^{x-1} \leq 2$ for all $0 < x < 1$. Hence,

$$\sigma_n \leq \sqrt{2} d_m^{\frac{n-m}{n}}$$

for all $1 \leq m < n$ follows. In particular, for $n = 2n'$ and $m = n'$ we get

$$\sigma_{2n'} \leq \sqrt{2d_{n'}}.$$

□

Now we apply the acquired results of this section to the setting of theorem 4.9.

Proof of theorem 4.9. (a) We will prove the claim by contradiction. Assume that $d_n \leq C_0 n^{-\alpha}$ for all $n \in \mathbb{N}$ and that there exists a $N \in \mathbb{N}$

$$\sigma_N > C_1 N^{-\alpha}$$

where $C_1 := 2^{5\alpha+1} C_0$. Denote by M the smallest integer such that

$$\sigma_M > C_1 M^{-\alpha}.$$

First, consider the case that $M = 4s$ for $s \in \mathbb{N}$.

Now apply theorem 4.11 to $N = K = n$ and any $1 \leq m < n$. Using σ_n monotonicity, we get

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

For $n = 4s$, $m = s$ this reduces to

$$\sigma_{4s} \leq \sqrt{2\sigma_{2s} d_s}.$$

Using the assumption $\sigma_{2s} \leq C_1(2s)^{-\alpha}$ and $d_s \leq C_0 s^{-\alpha}$ we obtain

$$\sigma_M \leq \sqrt{2^{1-\alpha} C_0 C_1} s^{-\alpha}.$$

However, by assumption

$$C_1(4s)^{-\alpha} < \sigma_M \leq \sqrt{2^{1-\alpha} C_0 C_1} s^{-\alpha}$$

and therefore

$$C_1 < 2^{1+3\alpha} C_0 < 2^{1+5\alpha} C_0 =: C_1$$

by definition of C_1. This contradiction yields (a) in the case that $M = 4s$.

However, if $M = 4s + q$ for $q \in \{1, 2, 3\}$, we have

$$\sigma_M > C_1(4s+q)^{-\alpha} > C_1 2^{-\alpha}(4s)^{-\alpha} = C_1 2^{-3\alpha} s^{-\alpha}$$

on the one hand and as σ_n is monotonic and we can use the previous case

$$\sigma_M = \sigma_{4s+q} \leq \sigma_{4s} \leq \sqrt{2^{1-\alpha} C_0 C_1} s^{-\alpha}$$

on the other hand. This gives the desired contradiction

$$C_1 < 2^{5\alpha+1} C_0 =: C_1.$$

(b) Let $d_n \leq C_0 e^{-c_0 n^\alpha}$ for all $n \in \mathbb{N}$. Define $c_1 := 2^{-1-2\alpha} c_0$. For the case that $n = 2s$ we apply corollary 4.12

$$\sigma_n \leq \sqrt{2C_0} e^{-c_0 2^{-1-\alpha}(2s)^\alpha} \leq \sqrt{2C_0} e^{-c_1 n^\alpha}$$

The case that $n = 2s + 1$ follows from the previous consideration:

$$\sigma_n \leq \sigma_{2s} \leq \sqrt{2C_0} e^{-c_0 2^{-1-\alpha}(2s)^\alpha} \leq \sqrt{2C_0} e^{-c_0 2_1^c (4s)^\alpha} \leq \sqrt{2C_0} e^{-c_1 n^\alpha}.$$

□

Remark. The estimates in this section can be composed respectively for a weaker version of these greedy algorithms: For a fixed parameter $0 < \gamma \leq 1$, choose $f_1 \in \mathcal{F}$ such that

$$\|f_1\| \geq \gamma \sigma_0 = \gamma \max_{f \in \mathcal{F}} \|f\|$$

and once $f_1, ..., f_{n-1}$ are chosen, choose $f_n \in \mathcal{F}$ such that

$$\sigma_n(f_n) := \|f_n - \Pi_{V_n} f_n\| \geq \gamma \sigma_n.$$

The estimates for this version differ by an exponent of γ^{-1} from our results.

4.2.3. Resulting Convergence Rates for the P-Greedy Algorithm

For $\mathcal{H}$ a reproducing kernel Hilbert space $\mathcal{H}_K(\Omega)$ and $\mathcal{F}$ the compact subset $\mathcal{V}(\tilde{\Omega})$ (see corollary 4.8), the greedy algorithm described in subsection 4.2.2 is exactly our P-greedy algorithm as introduced in section 3.5. It chooses the kernel translates $f_1 = K(\cdot, x_{j_1}), ..., f_n = K(\cdot, x_{j_n})$ as a basis of $V_n = V(X_n)$: First, f_1 is chosen such that

$$\begin{aligned} f_1 &= \operatorname*{arg\,max}_{f \in \mathcal{V}(\tilde{\Omega})} \|f\| = \operatorname*{arg\,max}_{K(\cdot,x) \in \mathcal{V}(\tilde{\Omega})} \|K(\cdot, x)\| \\ &= \operatorname*{arg\,max}_{K(\cdot,x) \in \mathcal{V}(\tilde{\Omega})} \sqrt{K(x,x)} = \operatorname*{arg\,max}_{K(\cdot,x) \in \mathcal{V}(\tilde{\Omega})} P_{V(X_0)}(x) = K(\cdot, x_{j_1}). \end{aligned}$$

For previously chosen $f_1 = K(\cdot, x_{j_1}), ..., f_n = K(\cdot, x_{j_n})$, we have $V_n = \operatorname{span}\{f_1, ..., f_n\} = \operatorname{span}\{K(\cdot, x_{j_1}), ..., K(\cdot, x_{j_n})\} = V(X_n)$. Choose

$$\begin{aligned} f_{n+1} &= \operatorname*{arg\,max}_{f \in \mathcal{V}(\tilde{\Omega})} E(f, V_n) \\ &= \operatorname*{arg\,max}_{K(\cdot,x) \in \mathcal{V}(\tilde{\Omega})} \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\| \\ &= \operatorname*{arg\,max}_{K(\cdot,x) \in \mathcal{V}(\tilde{\Omega})} P_{V(X_n)}(x) \\ &= K(\cdot, x_{j_{n+1}}). \end{aligned}$$

Theorem 4.9 thus gives us an estimate for $\sigma_n = \left\| P_{V(X_n)} \right\|_{L_\infty(\tilde{\Omega})}$:

$$\begin{aligned} \sigma_n &= E(\mathcal{V}(\tilde{\Omega}), V(X_n)) \\ &= \sup_{x \in \tilde{\Omega}} \left\| K(\cdot, x) - \Pi_{V(X_n)}(K(\cdot, x)) \right\| \\ &= \sup_{x \in \tilde{\Omega}} P_{V(X_n)}(x) \\ &= \left\| P_{V(X_n)} \right\|_{L_\infty(\tilde{\Omega})}. \end{aligned}$$

As seen in corollary 4.8, the P-greedy algorithm produces point sets X_n such that

(1) for K of finite smoothness $\beta \in \mathbb{N}$:

$$d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) \leq c_1 n^{-\frac{\beta}{d} + \frac{1}{2}}.$$

(2) and for K of infinite smoothness:

$$d_n(\mathcal{V}(\tilde{\Omega}), \mathcal{H}_K(\Omega)) \leq c_2 e^{-c_3 n^{1/d}}.$$

We can now deduce the following convergence result for our P-greedy algorithm.

Theorem 4.13. *Let K be a translational invariant kernel and $\Omega \subset \mathbb{R}^d$ satisfying an interior cone condition. For any $\tilde{\Omega} \subset \Omega$, the P-greedy algorithm chooses point sets $X_n \subset \tilde{\Omega}$ such that the following convergence rates hold:*

(1) For K of finite smoothness $\beta \in \mathbb{N}$:

$$\left\|P_{V(X_n)}\right\|_{L_\infty(\tilde{\Omega})} \leq d_1 n^{-\frac{\beta}{d}+\frac{1}{2}}.$$

(2) For K of infinite smoothness:

$$\left\|P_{V(X_n)}\right\|_{L_\infty(\tilde{\Omega})} \leq d_2 e^{-d_3 n^{1/d}}.$$

where $d_1 := 2^{\frac{5d}{\beta}-\frac{3}{2}} c_1, d_2 := \sqrt{2c_2}, d_3 := 2^{-1-\frac{2}{d}} c_3$.

Proof. Apply theorem 4.9 with $\alpha = \frac{\beta}{d} - \frac{1}{2}$ in the first case and $\alpha = \frac{1}{d}$ in the second. □

As we discussed in section 4.1, these are the best known convergence rates for any (in general non-greedy) point selections. The P-greedy algorithm thus produces point selections that are near-optimal.

4.3. Conclusion to the Point Distribution

In [5] the authors noticed that the P-greedy algorithm produces point sets X_n that are asymptotically uniform inside Ω. They proved the following theorem.

Theorem 4.14. *Let K be of finite smoothness β, see section 4.1. Let Ω satisfy an interior cone condition. Then for $\alpha > \beta$, there exists a constant M_α satisfying the following property: If $\epsilon_n > 0$ and $X_n \subset \Omega$ are given such that*

$$\left\|f - \Pi_{V(X_n)}(f)\right\|_{L_\infty(\Omega)} \leq \epsilon_n \|f\|$$

for all $f \in \mathcal{H}_K(\Omega)$, then

$$h_{X_n,\Omega} \leq M_\alpha \epsilon_n^{1/(\alpha-\frac{d}{2})}.$$

Yet, they could not prove the observed point distribution with their convergence result, theorem 4.4. However, we can with our improved bounds from theorem 4.13 for the case of kernels with finite smoothness β.

Corollary 4.15. *Assume K and Ω to be as specified in 4.14. Then there exists a constant $c > 0$ such that the point sets X_n selected by the P-greedy algorithm satisfy for any $\epsilon > 0$*

$$h_{X_n,\Omega} \leq c n^{-\frac{1}{d}(1-\epsilon)}$$

where c is independent of n.

Proof. Let $\epsilon > 0$. From our convergence result in theorem 4.13, we know that

$$\left\|P_{V(X_n)}\right\|_{L_\infty(\Omega)} \leq \hat{c_1} n^{-\frac{\beta}{d}+\frac{1}{2}} =: \epsilon_n.$$

According to theorem 4.14, for every $\alpha > \beta$ there exists a constant M_α such that

$$h_{X_n,\Omega} \leq M_\alpha \epsilon_n^{\frac{1}{\alpha - d/2}} = c n^{-\frac{1}{d}(\frac{\beta - d/2}{\alpha - d/2})}$$

with $c := M_\alpha \hat{c_1}^{\frac{1}{\alpha - d/2}}$. Since one can always find an $\alpha > \beta$ such that

$$\frac{\beta - d/2}{\alpha - d/2} = 1 - \epsilon,$$

the claim follows. □

CHAPTER 5

Implementing the P-Greedy Algorithm

Consider a target model $f : \Omega \to \mathbb{R}^q$ for $\Omega \subset \mathbb{R}^d$. For a fixed strictly positive definite kernel $K : \Omega \times \Omega \to \mathbb{R}$ and given data sites $\tilde{\Omega} := \{x_1, ..., x_N\} \subset \Omega$, we now want to implement the P-greedy algorithm that iteratively chooses point sets

$$X_k := \{x_{j_1}, ..., x_{j_k}\} = X_{k-1} \cup \{x_{j_k}\}$$

from $\tilde{\Omega}$ by the selection rule

$$x_{j_k} = \underset{x \in \tilde{\Omega} \setminus X_{k-1}}{\arg\max}\ P_{V(X_{k-1})}(x), \quad k \geq 1$$

where $X_0 := \emptyset$ and $P_{V(X_0)}(x) = \|K(\cdot, x)\| = K(x, x)$. It approximates the kernel interpolant

$$s_f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_{j_i}), \quad x \in \Omega,\ \alpha_i \in \mathbb{R}^q,\ i = 1, ..., N.$$

We implement it by constructing a Newton basis $(v_1, ..., v_k)$ of the subspaces

$$V(X_k) := \text{span}\{K(\cdot, x_{j_1}), ..., K(\cdot, x_{j_k})\}$$

and iteratively computing the interpolants

$$f_k(x) = \sum_{i=1}^{k} c_i v_i(x) = f_{k-1}(x) + c_k v_k(x)$$

with respective coefficients $c_1, ..., c_k \in \mathbb{R}^q$. When the interpolant meets a given tolerance on the Power Function or residual or the maximum expansion size is reached, the algorithm stops. Once it stops, the coefficients $\alpha_1, ..., \alpha_n$ with respect to the kernel basis $(K(\cdot, x_{j_i}))_{i=1,...,n}$ can be computed via a transition matrix and the interpolant

$$f_n = \sum_{i=1}^{n} \alpha_i K(\cdot, x_{j_i})$$

is returned.

Input: interpolation data, training parameters
Output: training results
Compute the kernel matrix A_{K,X_N};
Initialize needed variables;
for $k = 1, ..., maxIter$ **do**
  Point selection x_{j_k} for this iteration;
  Compute the k-th basis function v_k;
  Compute the Power Function values $P_{V(X_k)}(x_i)$ for $i = 1, ..., N$;
  if *data dependent* **then**
    Compute the k-th coefficient c_k w.r.t. the Newton basis;
    Compute the residual $r_k = f - f_k$;
    Update the basis transition matrix;
    if $f \in \mathcal{H}_K(\Omega)$ **then**
      Compute $\|f - f_k\|_{\mathcal{H}_K(\Omega)}$;
    end
  end
  if $\max_{x \in X_N}\{P_{V(X_k)}(x)\} < tolerance$ *or* $\max_{x \in X_N}\{r_k(x)\} < tolerance$ **then**
    Save the number k of needed iterations as n;
    exit for-loop;
  end
end
if *data dependent* **then**
  Compute the resulting approximation values $f_n(x_1), ..., f_n(x_N)$;
  Compute the coefficients $\alpha_1, ..., \alpha_n$ w.r.t. the kernel basis;
end
return *approximation and training results*

Algorithm 1: Implementation of P-Greedy Algorithm

5.1. The Implementation

Algorithm 1 displays our implementation in python. We implemented the algorithm as a method `train(interpolation_data, train_param)` of the python script `pgreedy.py`. The entire code can be found in the appendix.

The method expects python dictionaries `interpolation_data` and `train_param` as input.

`interpolation_data` needs to contain

`data`: A 2d-array of shape (N, d). It contains the available data sites $x_1, ..., x_N$.

Since the P-greedy algorithm is data independent, we implemented the algorithm such that it is also possible to pass `interpolation_data` without values of a specific function f. In that case, only the point selection, Newton basis and the Power Function are computed. Hence, `interpolation_data` can also contain

`f`: A 2d-array storing the respective function values $f(x_1), ..., f(x_N)$. It is of shape (N, q).

`rkhs_norm_2`: $\|f\|^2_{\mathcal{N}_K(\Omega)^q}$. If it is passed, the RKHS-error $\|f - f_k\|^2_{\mathcal{N}_K(\Omega)^q}$ is computed in the k-th iteration.

`train_param` needs to contain:

`kernel`: A strictly positive definite kernel $K : \Omega \times \Omega \to \mathbb{R}$.

Moreover, it can contain the following:

`max_iterations`: The maximum number of iterations, i.e. maximum expansion size of the interpolants. If it is not declared, it is set to N.

`p_tolerance`: A tolerance value on the Power Function.

`r_tolerance`: A tolerance value on the residual.

`train(interpolation_data, train_param)` returns a dictionary `results`. It contains the following:

`selected`: The indices $j_1, ..., j_n$ of the final point selection X_n.

`num_iterations`: The number of iterations n, i.e. expansion size.

`max_power_fct`: The maximum value of $P_{V(X_k)}(x)$ over each $x \in \tilde{\Omega}$ for each iteration k.

Moreover, if f was passed, `results` also holds

`surrogate`: A 2d-array of shape (N, q) that contains the surrogate values $f_n(x_1), ..., f_n(x_N)$.

`kernel_coeff`: A 2d-array of shape (n, q) that contains the kernel coefficients $\alpha_1, ..., \alpha_n$.

`max_residual`: An 1d-array that stores the maximum value of $\|r_k(x)\|_2$ over each $x \in \tilde{\Omega}$ for each iteration k.

Also, if $\|f\|_{\mathcal{N}_K(\Omega)^q}$ was passed:

`rkhs_error`: An 1d-array storing the interpolation error in RKHS-norm $\|f - f_k\|^2_{\mathcal{N}_K(\Omega)^q}$ for each iteration k.

We will now describe some details of the implementation.

Since the number of available data N can be very large, we avoid computing the entire kernel matrix A_{K,X_N} by defining a function that returns a submatrix of A_{K,X_N} for given indices of X_N when needed.

```
# function returning kernel values of data for given indices
def kernel_matrix(k,l):
  if data.ndim > 1:
    return kernel(data[k,:], data[l,:])
  else:
    return kernel(data[k], data[l])
# function returning a submatrix of the kernel matrix on data
kernel_matrix_vectorized = np.vectorize(kernel_matrix)
# diagonal of kernel matrix
kernel_matrix_diagonal = np.array([kernel_matrix(i,i) for i in range(num_data)])
```

Listing 5.1: Kernel Matrix

Consider the k-th iteration. First, the point selection is done according to

$$x_{j_k} = \underset{x \in \tilde{\Omega} \setminus X_{k-1}}{\arg\max} P_{V(X_{k-1})}(x), \quad k \geq 1.$$

```
if k > 0:
  selection_index = np.argmax(power_fct[notselected,k-1])
else:
  selection_index = np.argmax(kernel_matrix_diagonal)
selected.append(notselected[selection_index])
```

Listing 5.2: Point Selection

The k-th function v_k of the Newton basis is computed according to the characterization we have seen in section 3.5, equation (3.7):

$$v_1 = \frac{K(\cdot, x_{j_1})}{\sqrt{K(x_{j_1}, x_{j_1})}} \qquad \text{and}$$
$$v_k = \frac{K(\cdot, x_{j_k}) - \sum_{i=1}^{k-1} v_i(x_{j_k}) v_i}{P_{V(X_{k-1})}(x_{j_k})}, \quad k \geq 2.$$

```
if k > 0:
  a = newton_basis[notselected, 0:k].T
  b = newton_basis[selected[k], 0:k]
  kernel_values = kernel_matrix_vectorized(notselected, selected[k]).ravel()
  newton_basis[notselected, k] =  kernel_values - b @ a
  newton_basis[notselected, k] /= power_fct[selected[k], k-1]
else:
  kernel_values = kernel_matrix_vectorized(notselected, selected[k]).ravel()
  power = np.sqrt(kernel_matrix_diagonal[k])
  newton_basis[notselected, k] = kernel_values / power
```

Listing 5.3: Computing Newton basis

We only have to compute the Newton basis on `notselected` since we initialized `newton_basis` with zeros and $v_k(x_m) = 0$ for $m = 1, ..., k-1$:

$$\begin{aligned} v_k'(x_m) &= K(x_m, x_{j_k}) - \sum_{i=1}^{k-1} \langle K(\cdot, x_{j_k}), v_i \rangle v_i(x_m) \\ &= K(x_m, x_{j_k}) - \sum_{i=1}^{k-1} \langle K(\cdot, x_m), v_i \rangle v_i(x_k) \\ &= K(x_m, x_{j_k}) - \Pi_{V(X_{k-1})}(K(\cdot, x_m))(x_{j_k}) \\ &= 0, \end{aligned}$$

since $K(\cdot, x_m) \in V(X_{k-1})$ for $m = 1, ..., k-1$. Since `notselected` holds the current selection index k until the end of the k-th iteration, $v_k(x_k)$ is also computed.

We compute the Power function according to corollary 3.7:

$$P_{V(X_k)}(x)^2 = P_{V(X_{k-1})}(x)^2 - v_k(x)^2, \quad k \geq 1.$$

```
if k > 0:
  power_squared = power_fct[notselected, k-1]**2
else:
  power_squared = kernel_matrix_diagonal[notselected]
basis_squared = newton_basis[notselected, k]**2
power_fct[notselected, k] = np.sqrt(np.abs(power_squared - basis_squared))
max_power_fct[k] = np.max(power_fct[:,k])
```

Listing 5.4: Computing the Power Function

We only have to compute the Power function on `notselected` since $P_{V(X_k)}$ vanishes on X_k, see corollary 3.6.

Now, assume that function values `f` were passed. We compute the k-th coefficients c_k with respect to the Newton basis according to proposition 3.9:

$$c_k = \frac{r_{k-1}(c_{j_k})}{P_{V(X_{k-1})}(x_{j_k})}, \quad k \geq 1.$$

```
if k > 0:
  r = residual[k-1, selected[k], :]
   p = power_fct[selected[k], k-1]
   newton_coeff[k, :] = r / p
else:
   power = np.sqrt(kernel_matrix(selected[k],selected[k]))
   newton_coeff[k, :] = f[selected[k], :] / power
```

Listing 5.5: Computing the Newton Coefficients

Then, we compute the residuals

$$r_k(x_i) = r_{k-1}(x_i) - c_k v_k(x_i), \quad i = 1, ..., N$$

according to equation (3.8), with $r_0 := f$.

```
if k > 0:
  r = residual[k-1, :, :]
else:
  r = f
cv = np.outer(newton_basis[:, k], newton_coeff[k, :])
residual[k, :, :] = r - cv
max_residual[k] = np.max(np.linalg.norm(residual[k,:,:], axis=1))
```

Listing 5.6: Computing the Residual

Here, `residual` is a 3d-array, where the first axis matches the iteration, the second the data value and the third the output component.

Next, we compute the transition matrix $(a_{ij})_{i,j=1,...,n}$ for the change of basis according to Proposition 3.10:

$$\begin{aligned} a_{ik} &= -\frac{1}{P_{V(X_{k-1})}(x_{j_k})} \sum_{l=1}^{k-1} a_{il} v_k(x_{j_k}), && i = 1, ..., k-1, \\ a_{ik} &= \frac{1}{P_{V(X_{k-1})}(x_{j_k})}, && i = k, \\ a_{ik} &= 0, && i = k+1, ..., n. \end{aligned}$$

```
t = transition_matrix[0:k, 0:k]
n = newton_basis[selected[k], 0:k].T
transition_matrix[0:k, k] = - t @ n
transition_matrix[k, k] = 1
if k > 0:
  transition_matrix[:, k] /= power_fct[selected[k], k-1]
else:
  transition_matrix[:, k] /= kernel_matrix(selected[0], selected[0])
```

Listing 5.7: Computing the Transition Matrix

`transition_matrix` was initialized with zeros.

Lastly, if $\|f\|^2_{\mathcal{N}_K(\Omega)^q}$ was passed, the $\|\cdot\|_{\mathcal{N}_K(\Omega)^q}$-error is computed according to equation 3.10:

$$\|f - f_k\|^2_{\mathcal{N}_K(\Omega)^q} = \sum_{i=1}^{q} \left\|f^{(i)}\right\|^2_{\mathcal{N}_K(\Omega)} - \left\|f_k^{(i)}\right\|^2_{\mathcal{N}_K(\Omega)}.$$

In every iteration, we compute $\sum_{i=1}^{q} \left\|f_k^{(i)}\right\|^2_{\mathcal{N}_K(\Omega)}$ by

```
rkhs_norm_surrogate[k] = np.sum(newton_coeff[0:k+1, :]**2)
```

Listing 5.8: Computing the RKHS-Norm of the Surrogates

Once the algorithm has stopped, $\|f - f_i\|^2_{\mathcal{N}_K(\Omega)^q} = \|f\|^2_{\mathcal{N}_K(\Omega)^q} - \|f_i\|^2_{\mathcal{N}_K(\Omega)^q}$ for each iteration $i = 1, ..., n$ is computed by

```
rkhs_error = norm_squarred - rkhs_norm_surrogate
```

Listing 5.9: Computing the Squarred RKHS-Error

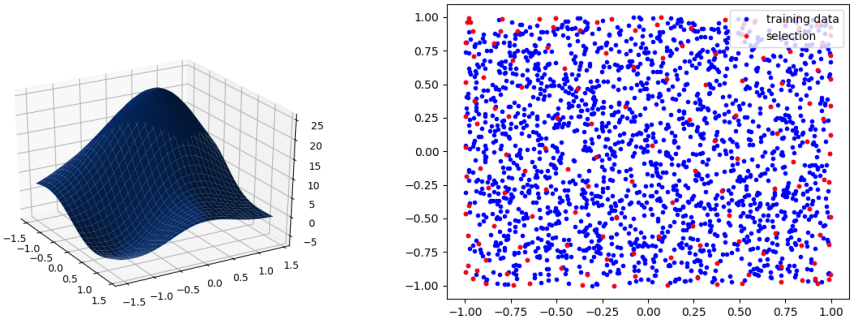

Figure 5.1.: Target Function f and Training Data $x_1, ..., x_N$

At the end of the k-th iteration, it is checked if f_k meets the (possibly) given tolerances. If it does, k is saved as the number of needed iterations and all variables that have space for more iterations are cut off at iteration k.

Then, the training results are saved in the dict `results` and and the following training information is printed:

`number of training data sites`: number N of available data sites

`number of selected data sites`: the number n of selected data sites $x_{j_1}, ..., x_{j_n}$, i.e. the expansion size of f_n

`max of power function on training data`: $\max_{1 \leq i \leq N} P_{V(X_n)}(x_i)$.

If f was passed, also

`mean residual on training data`: the arithmetic mean of $r_n(x_1), ..., r_n(x_N)$

`max residual on training data`: the maximum value of $r_n(x_1), ..., r_n(x_N)$

and if $\|f\|^2_{\mathcal{N}_K(\Omega)^q}$ was passed,

`max RKHS error`: $\|f - f_n\|^2_{\mathcal{N}_K(\Omega)^q}$

is printed.

5.2. Example On Artificial Data

Just as in [2], we are trying out our interpolation algorithm in an artificial setting as a first example. Let's consider the domain

$$\Omega = [-1, 1]^2 \subset \mathbb{R}^2$$

and draw $N = 2000$ random samples as training data $x_1, ..., x_N$ from it. We fix the Gaussian kernel

$$K(x, y) = e^{-\|x-y\|_2^2}, \quad x, y \in \mathbb{R}^2$$

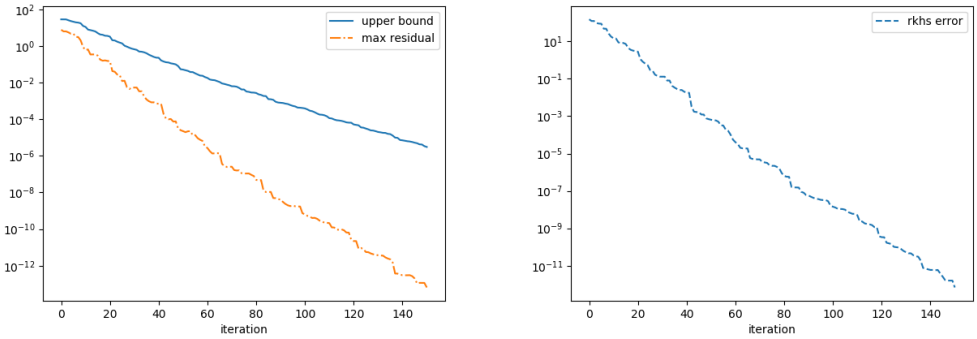

Figure 5.2.: $\max_{i=1,\dots,N} \|r_n(x_i)\|_2$ (left) and $\|f - f_n\|^2_{\mathcal{N}_K(\Omega)^q}$ (right) at each Iteration n

with shaping parameter 1 and choose an easy target function, namely a linear combination of $m = 100$ kernel translates

$$f = \sum_{i=1}^{m} \beta_i K(\cdot, z_i) : \mathbb{R}^2 \to \mathbb{R}$$

for randomly chosen centers $z_1, ..., z_m \in \Omega$ and randomly chosen coefficients

$$\beta_1, ..., \beta_m \in [-5, 5] \subset \mathbb{R}.$$

Figure 5.1 depicts an example of this target function f.

As training parameters, we set the maximum expansion size to $n_{max} = 190$ and set the tolerance on

$$\max_{i=1,\dots,N} P_{V(X_n)}(x_i) \quad \text{and} \quad \max_{i=1,\dots,N} \|r_n(x_i)\|_2$$

to 10^{-13} for $n \leq n_{max}$.

```
Training results:

number of training data sites: 2000
number of selected data sites/expansion size: 151
max of power function on training data: 1.0726531321979136e-07
mean residual on training data: 1.1788403149132925e-14
max residual on training data: 6.465420936777982e-14
max RKHS error: 6.821210263296962e-13
```

Listing 5.10: Training Output

Listing 5.10 shows our training results. The P-greedy algorithm stopped at iteration 151 since the maximum residual

$$\max_{i=1,\dots,N} \|r_{151}(x_i)\|_2 = 6.47 \cdot 10^{-14}$$

met the passed threshold of 10^{-13}. Figure 5.2 shows the semi-log plot of the residual and the approximation error in RKHS-norm. On the left, it illustrates the development of r_n and its upper bound

$$r_n \leq \left\| P_{V(X_n)} \right\|_{L_\infty(\Omega)} \|f\| \quad \forall n \in \mathbb{N}.$$

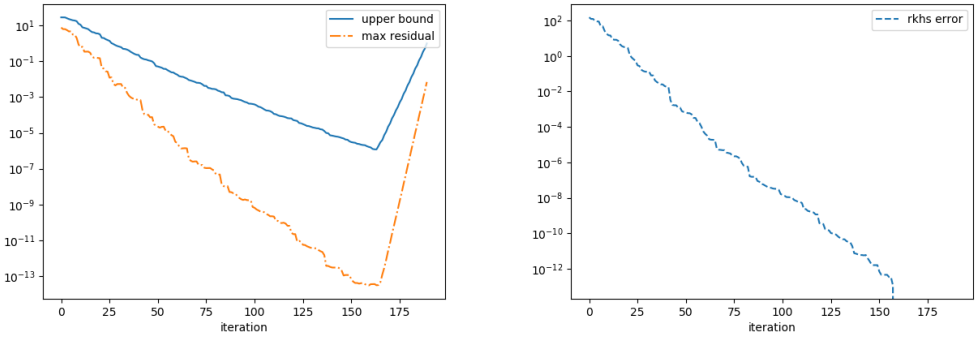

Figure 5.3.: Numerical Instability after Iteration 159

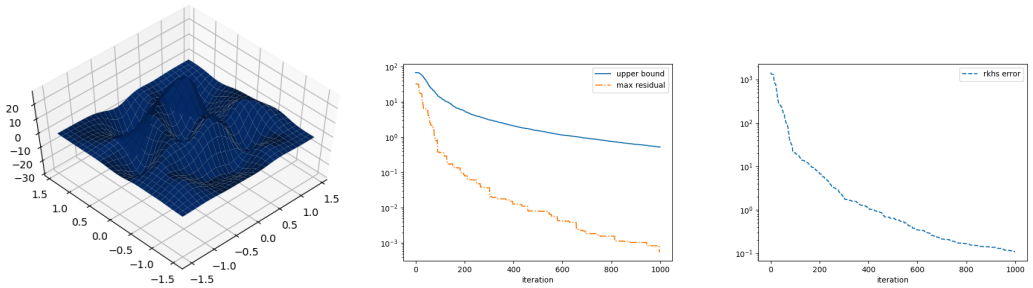

Figure 5.4.: Target Function f and Training Results for Wendland Kernel $K_{\Phi_{2,2}}$

As it shows exponential decay of the Power Function, it confirms our convergence result from chapter 4. On the right side, figure 5.2 illustrates the development of $\|f - f_n\|^2_{\mathcal{N}_K(\Omega)^q}$ over each iteration n. The approximation error in $\|\cdot\|_{\mathcal{N}_K(\Omega)^q}$ also decays exponentially.

However, when we do not set a tolerance on $P_{V(X_n)}$ and r_n, the algorithm continues up to an expansion size of $n_{max} = 190$ as seen in the training output in listing 5.11. In this case, we observe numerical instability after iteration 159 as the Power Function has almost reached single machine precision. We are considering single precision since we are taking the square root when computing the Power Function.

```
Training results:

number of training data sites: 2000
number of selected data sites/expansion size: 190
max of power function on training data: 0.034922219448066149
mean residual on training data: 0.0015198165286859368
max residual on training data: 0.006958019580380063
max RKHS error: -0.021370832900061032
min residual of 2.992602247649948e-14 at iteration 159
min power of 4.2112472332153586e-08 at iteration 163
min rkhs error of -0.021370832900061032 at iteration 189
```

Listing 5.11: Results for Training up to Maximum Expansion Size

On the other hand, let us now consider the Wendland Kernel

$$K_{\Phi_{2,2}}(x,y) \doteq (1 - \|x-y\|_2)^6(35\|x-y\|_2^2 + 18\|x-y\|_2 + 3), \quad x,y \in \mathbb{R}^2.$$

of smoothness C^4. Analogously to the above example, we choose $\Omega = [-1,1]^2$ and the target function to be

$$f = \sum_{i=1}^{m} \beta_i K_{\Phi_{2,2}}(\cdot, z_i) : \mathbb{R}^2 \to \mathbb{R}$$

for $m = 100$ randomly chosen centers $z_1, ..., z_m \in [-1,1]^2$ and randomly chosen coefficients $\beta_1, ..., \beta_m \in [-5,5]^2$. Listing 5.12 and figure 5.4 show the training results. The interpolation converges much slower than for the Gaussian kernels.

```
Training results:

number of training data sites: 2000
number of selected data sites/expansion size: 1000
max of power function on training data: 0.013541104026304677
mean residual on training data: 2.757516617933897e-05
max residual on training data: 0.0005660627971627867
max RKHS error: 0.10663536831316378
```

Listing 5.12: Results for Training With Wendland Kernel

CHAPTER 6

Application of the P-greedy Algorithm

We will now discuss a specific application of our implemented P-greedy algorithm. Consider the MNIST data base of handwritten digits from [4], see figure 6.1. It contains 60,000 training and 10,000 test examples of labeled 28×28 pixel black and white images of handwritten digits from 0 to 9. Our goal in this application is to predict the corresponding digit label for each input image.

6.1. Convolutional Neural Network Implemented in LeNet-5 Architecture

This is the standard example of a classification task that is nowadays approached by a convolutional neural network. LeNet-5 is a specific architecture of such a convolutional neural network and handles this classification task very well. It was introduced by Yann LeCun in [3] who is one of the pioneers in the work with convolutional neural networks.

Figure 6.2 shows the network architecture. It takes the 28×28 pixel images as input and turns it into 32×32 pixel images by adding zero padding. The first convolutional layer applys 6 convolutional filters of size 5×5 with stride 1 so that it produces 6 images of size 28×28 per input image. The second layer then performs max pooling of size 2×2 and stride 2 so that it reduces the images to 6 images of size 14×14. Again, a convolutional layer with 16 filters of size 5×5 (this time without zero padding) with stride 1 and a max pooling layer are applied. They turn the training values into 16 images of size 5×5. One

Figure 6.1.: MNIST Data Base (image from [15])

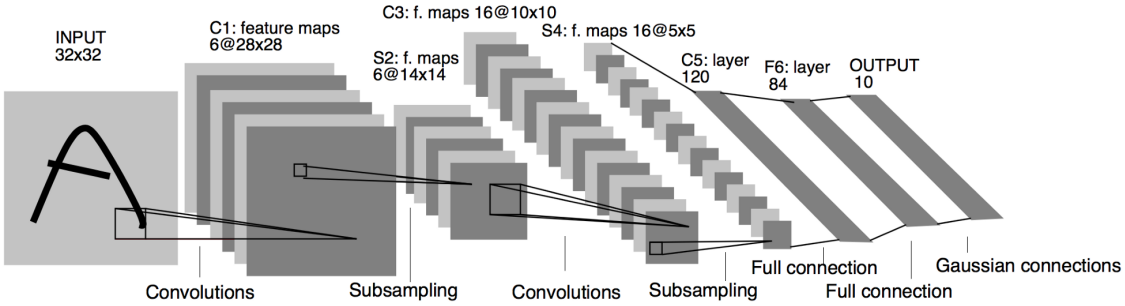

Figure 6.2.: LeNet-5 Architecture (image from [3])

last convolutional layer applies 120 filters of size 5×5 and turns the images into an array of 120 values. These are the so-called *feature vectors*. Two dense layers, the first with 84, the second with 10 nodes, then perform the classification and produce a vector with 10 probabilities - one for each possible digit - by computing the softmax. An implementation of the network using tensorflow can be found in the appendix. Listing 6.1 captures the training output. The network achieved an accuracy of 98.53 % on the 60.000 training images and an accuracy of 98.96 % on the 10.000 test images.

```
Using TensorFlow backend.
Saving output to output.npy...
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [==============================] - 20s 327us/step - loss: 0.2683 -
    acc: 0.9146 - val_loss: 0.0559 - val_acc: 0.9824
Epoch 2/3
60000/60000 [==============================] - 21s 342us/step - loss: 0.0641 -
    acc: 0.9803 - val_loss: 0.0484 - val_acc: 0.9845
Epoch 3/3
60000/60000 [==============================] - 20s 335us/step - loss: 0.0464 -
    acc: 0.9853 - val_loss: 0.0319 - val_acc: 0.9896
Lenet Test Loss: 0.03187649185821647
Lenet Test Accuracy: 0.9896
```

Listing 6.1: Training Output of Neural Network

6.2. Classifying MNIST Data with P-Greedy Algorithm

Let's now see how to apply our implemented P-greedy algorithm in this setting. We are interested in the question how well the P-greedy algorithm manages the classification task. The task can be seen as finding a function f that determines a vector with 10 probabilities per input image, one for each digit. It can thus be formulated formulated as a high-dimensional interpolation problem where the 10-dimensional output vectors indicate the probabilities that the input image depicts digit 0,1,...,9.

We will use our implementaion of the convolutional neural network as preprocessing of the input data and perform the P-greedy algorithm on the 120-dimensional feature vectors that arise after the last convolutional layer. Hence, we can see how well the interpolation procedure manages the classification task compared to the two dense layers in the LeNet-5

architecture.

Let's be more precise on how we formulate the interpolaton problem. The neural network can be considered as a model

$$N : [0,1]^{32\times 32} \to \mathbb{R}^{120}$$

that determines a feature vector for every input image and our target function as

$$f : \Omega \to \mathbb{R}^{10},$$

where $\Omega := N([0,1]^{32\times 32}) \subset \mathbb{R}^{120}$, that determines a vector of probabilities, one for each digit. We are looking for an interpolant f_n according to the P-greedy selection $x_{j_1}, ..., x_{j_n} \in \Omega$ that provides a good approximation:

$$f_n = \sum_{i=1}^{n} \alpha_i K(\cdot, x_{j_i})$$

for corresponding coefficients $\alpha_1, ..., \alpha_n \in \mathbb{R}^{10}$. We sample $x_1, ..., x_N$ pairwise distinct data points from the set of all 60.000 feature vectors, for $N \leq 60,000$. Let $k_i \in \{0, 1, ..., 9\}$ denote the digit corresponding to feature vector x_i, $i = 1, ..., N$. The interpolation conditions can then be formulated as

$$f_n(x_i) = f(x_i), \quad i = 1, ..., N$$

where

$$f(x_i) = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{10}$$

is the vector that is 0 in every component except at position k_i where it is 1.

We will use a Gaussian kernel. However, because of quite large values in $\|x_i - x_j\|_2^2$, $i, j \in \{1, ..., N\}$, we choose the scaling parameter to be $\rho = 1/100$. We are thus using the strictly positive definite kernel

$$K(x, y) = e^{-\rho\|x-y\|_2^2}, \quad x, y \in \mathbb{R}^{120}$$

on $\mathbb{R}^{120}$. We perform the interpolation on the entire training set of 60,000 training data $x_1, ..., x_{N_{train}}$ and we choose a maximum expansion size of $N_{max} = 4000$.

Our training results for the P-greedy algorithm can be seen in figure 6.3. One can see that the Power Function is decreasing much slower than in the artificial example, yet it is decreasing steady. The mean residual decays as expected. However, the maximum residual seems to be constantly a little bit over 1. This can be explained as follows. Since we can not know how the neural network determines the feature vectors, we do not know much about our target function. Hence, we can not assume continuity of f. When two data sites are similar in $\|\cdot\|_2$, but represent different digits, the algorithm will try to classify them identically and tries to produce a 0 where a 1 is supposed to be and 1 where a 0 is supposed to be for one of the two. This misclassification will result in an approximation error > 1.

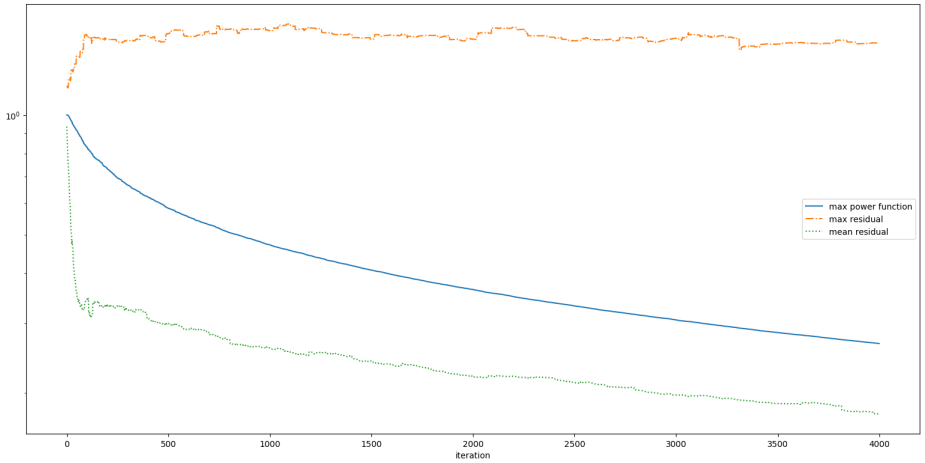

Figure 6.3.: Training Result of Application for P-Greedy Algorithm

To compute the accuracy of our algorithm, we apply our resulting surrogate f_n to the 10,000 test data $x_1, ..., x_{N_{test}}$ and choose the index with the highest value in $f_n(x_i)$, $i = 1, ..., x_{N_{test}}$, as our predicted digit. The accuracy is the ratio of correct predictions on the test data set. As seen in listing 6.2, we reach an accuracy of 99.37 % on the 60,000 training data samples and an accuracy of 98.87 % on the 10,000 test data samples. The test accuracy is thus almost just as good as the test accuracy of 98.96 % when classifying with the dense layers in the convolutional neural network.

```
Training results:

number of training data sites: 60000
number of selected data sites/expansion size: 4000
max of power function on training data: 0.26638571769193353
mean residual on training data: 0.17709936875201213
max residual on training data: 1.5197630814091492
train accuracy:  0.99365
test accuracy:  0.9887
```

Listing 6.2: Training Output of P-Greedy Algorithm

CHAPTER 7

Conclusion

The aim of this thesis was to study an interface between numerical mathematics and learning theory. Kernels present such an interface as they play an important role in both fields. We studied the use of kernels in an interpolation problem and applied it to a classification problem in learning theory.

First, we discussed some theory on kernels and became familiar with the Gaussian and Wendland kernels as examples of strictly positive definite kernels. These kernels were very useful in our implementation of kernel interpolation. We saw that strictly positive definite kernels $K : \Omega \times \Omega \to \mathbb{R}$ induce a Hilbert space of functions $\mathcal{N}_K(\Omega)$, the Native Space. For example for Wendland kernels, this is a Sobolev space. In these Native Spaces every element $f \in \mathcal{N}_K(\Omega)$ can be reproduced by kernel translates as the Riesz representer:

$$\langle f, K(\cdot,)\rangle = f(x) \quad \forall x \in \Omega.$$

This was a useful tool for the interpolation procedure in $\mathcal{N}_K(\Omega)$ as the numeric computation of the $\mathcal{N}_K(\Omega)$-inner product could be circumvented by computationally cheap kernel evaluations.

This gave us a tool at hand to find a surrogate $s_f : \Omega \to \mathbb{R}^q$ of a target function $f : \Omega \to \mathbb{R}^q$, $\Omega \subset \mathbb{R}^d$, of the form

$$s_f = \sum_{i=1}^{N} \alpha_i K(\cdot, x_i)$$

that satisfies the interpolation conditions

$$s_f(x_i) = f(x_i), \quad \forall i = 1, ..., N$$

for some given data $(x_1, f(x_1)), ..., (x_N, f(x_N)) \in \Omega \times \mathbb{R}^q$. However, as solving the full linear system is ill-conditioned, we introduced different greedy techniques that sample a selection $x_{j_1}, ..., x_{j_n}$, $n << N$, from all available data sites and compute the surrogates

$$f_n = \sum_{i=1}^{n} \alpha_i K(\cdot, x_{j_i}).$$

The P-greedy algorithm is one of them. We found out that it is a convergent algorithm that lets the Power Function decay exponentially for Gaussian kernels. It even turned out that it is as good as the convergence rates for the best known (non-greedy) point distributions. Our implementation of the P-greedy algorithm worked very well on artificial data and confirmed our proven convergence rates.

Lastly, we transferred our study to an application in learning theory. We wanted to see how kernel interpolation could handle a standard classification task interpreted as a high-dimensional interpolation problem. Using a convolutional neural network as preprocessing of the input images, the P-greedy algorithm handled the classification task very well and was able to achieve a classification accuracy as good as with the neural network.

Bibliography

[1] R. DeVore, G. Petrova, and P. Wojtaszczyk. Greedy algortihms for reduced bases in banach spaces. *Constr. Approx., 37(3):455-466*, 2013.

[2] B. Haasdonk and G. Santin. Greedy kernel approximation for sparse surrogate modelling. *Keiper W., Milde A., Volkwein S. (eds) Reduced-Order Modeling (ROM) for Simulation and Optimization. Springer, Cham*, 2018.

[3] Y. LeCun, L. Bottoua, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[4] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

[5] S. D. Marchi, R. Schaback, and H. Wendland. Near-optimal data-independent point locations for radial basis function interpolation. *Adv. Comp. Math., 23(3):317-330*, 2005.

[6] S. Müller. *Komplexität und Stabilität von kernbasierten Rekonstruktionsmethoden*. PhD thesis, Georg-August-Universität Göttingen, 2008.

[7] S. Müller and R. Schaback. A newton basis for kernel spaces. *Journal of Approximation Theory*, 2009.

[8] M. Pazouki and R. Schaback. Bases for kernel-based spaces. *Journal of Computational and Applied Mathematics*, 2011.

[9] A. Pinkus. *N-Widths in Approximation Theory*. Springer Verlag, 1985.

[10] G. Santin and B. Haasdonk. Convergence rate of the data-independent p-greedy algorithm in kernel-based approximation. *arXiv 1612.02672, University of Stuttgart*, 2016. Submitted to Dolomites Research Notes on Approximation.

[11] R. Schaback. Error estimates and condition numbers for radial basis function interpolation. *Adv. Comput. Math., 3(3):251-264*, 1995.

[12] R. Schaback and H. Wendland. Kernel techniques: From machine learning to meshless methods. *Acta Numerica*, 2006.

[13] B. Schölkopf and A. J. Smola. *Learning With Kernels*. MIT Press, 2002.

[14] H. Schröder. *Funktionalanalysis*. Harri Deutsch, 2000.

[15] J. Steppan. A few samples from the mnist test dataset. `https://commons.wikimedia.org/wiki/File:MnistExamples.png`, Dec. 2017.

[16] H. Wendland. *Scattered Data Approximation*. Cambridge University Press, 2005.

[17] D. Werner. *Funktionalanalysis*. Springer Berlin Heidelberg, 2018.

APPENDIX A

Mathematical Preliminaries

A.1. The Fréchet-Riesz Theorem

The following proof can be found in [14].

Theorem A.1 (Fréchet-Riesz theorem)**.** *Let H be a Hilbert space and let $\phi \in H'$ be any continuous linear functional on H. Then there exists a unique $x_0 \in H$ such that $\phi = \langle \cdot, x_0 \rangle$ and $\|\phi\| = \|x_0\|$.*

Proof. Consider $M := \ker \phi$. Since ϕ is continuous, M is closed. Assume $M \neq H$. Then $M^\perp \neq \{0\}$ and we can find a $0 \neq y_0 \in M^\perp$ such that $\phi(y_0) = 1$. Since

$$\phi(x - \phi(x)y_0) = \phi(x) - \phi(x)\phi(y_0) = 0,$$

it follows that

$$x - \phi(x)y_0 \in M$$

and hence

$$0 = \langle x - \phi(x)y_0, y_0 \rangle = \langle x, y_0 \rangle - \phi(x) \|y_0\|^2 .$$

Choose $x_0 := \frac{y_0}{\|y_0\|^2}$. Then, for all $x \in H$

$$\langle x, x_0 \rangle = \frac{\langle x, x_0 \rangle}{\|y_0\|^2} = \phi(x).$$

holds. If $M = H$, then $\phi = 0$. In this case, choose $x_0 = 0$.

Let's prove that this choice is unique. Let $x_1 \in H$ be another element satisfying $\phi = \langle \cdot, x_1 \rangle$. Then, we get

$$0 = \langle x, x_0 \rangle - \langle x, x_1 \rangle = \langle x, x_0 - x_1 \rangle \quad \forall x \in H$$

and thus $x_0 - x_1 \in H^\perp = \{0\}$.

Furthermore, $\|\phi\| \leq \|x_0\|$ holds by the Cauchy-Schwartz inequality

$$|\phi(x)| = |\langle x, x_0 \rangle| \leq \|x\| \, \|x_0\|$$

and since

$$\left| \phi\left(\frac{x_0}{\|x_0\|}\right) \right| = \frac{\langle x_0, x_0 \rangle}{\|x_0\|} = \|x_0\| ,$$

$\|\phi\| = \|x_0\|$ follows. □

A.2. Necessary Fourier Theory

Here, we will cite the auxiliary tools from chapter 5 in [16].

Definition A.2. *For $f \in L_1(\mathbb{R}^d)$ the Fourier transform $\widehat{f}$ is defined by*

$$\widehat{f}(x) := (2\pi)^{-d/2} \int_{\mathbb{R}^d} f(\omega) e^{-ix^T\omega} d\omega$$

and the inverse Fourier transform by

$$f^\vee(x) := (2\pi)^{-d/2} \int_{\mathbb{R}^d} f(\omega) e^{ix^T\omega} d\omega.$$

Proposition A.3. *The Fourier transformation is a bounded linear operator on a dense subset of $L_2(\mathbb{R}^d)$. Therefore, there exists a unique bounded extension T of this transformation on $L_2(\mathbb{R}^d)$ which we will call the Fourier transformation on $L_2(\mathbb{R}^d)$. We will use the same notation $\widehat{f} = Tf$ for $f \in L_2(\mathbb{R}^d)$.*

Corollary A.4 (Plancherel)**.** *There exists an isomorphic mapping $T : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ such that the following properties hold:*

1. $\|Tf\|_{L_2(\mathbb{R}^d)} = \|f\|_{L_2(\mathbb{R}^d)}$ *for all* $f \in L_2(\mathbb{R}^d)$
2. $Tf = \widehat{f}$ *for all* $f \in L_2(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$
3. $T^{-1}g = g^\vee$ *for all* $g \in L_2(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$.

APPENDIX B

Implementations

B.1. Implemented P-Greedy Algorithm

```
import numpy as np

"""
To interpolate a given model f: Omega -> R^output_dim with the use of a kernel
basis, the p-greedy algorithm chooses kernel translates iteratively and computes
the interpolant in each iteration. Once a tolerance is met or the max number of
iterations is reached, the interpolant is computed and the algorithm stops.

We are assuming:
    - data is an 2d-np.array of shape (num_data, d) containing all
        available data sites in Omega \subset R^d
    - kernel is a strictly positive definite kernel from Omega x Omega to R
    - f is an 2d-np.array of shape (len(data), output_dim) containing the values
        of f evaluated on data
"""

def train(interpolation_data, train_param):
    # load interpolation data
    data = interpolation_data['data']
    num_data = data.shape[0]
    data_dependent = 'f' in interpolation_data
    f_is_rkhs = 'rkhs_norm_2' in interpolation_data
    if data_dependent:
        f = interpolation_data['f']
    if f_is_rkhs:
        norm_squarred = interpolation_data['rkhs_norm_2']

    # load training parameters
    kernel = train_param['kernel']
    if 'max_iterations' in train_param:
        max_iterations = train_param['max_iterations']
    else:
        max_iterations = num_data
    if 'p_tolerance' in train_param:
        p_tol = train_param['p_tolerance']
    else:
        p_tol = 0
    if 'r_tolerance' in train_param:
        r_tol = train_param['r_tolerance']
    else:
        r_tol = 0

    # function returning kernel values of data for given indices
    def kernel_matrix(k,l):
        if data.ndim > 1:
            return kernel(data[k,:], data[l,:])
        else:
```

```
            return kernel(data[k], data[l])
    # function returning a submatrix of the kernel matrix on data
    kernel_matrix_vectorized = np.vectorize(kernel_matrix)
    # diagonal of kernel matrix
    kernel_matrix_diagonal = np.array([kernel_matrix(i,i) for i in range(num_data
        )])

    # initializing needed variables
    # selected indices
    selected = []
    # not selected indices
    notselected = list(range(num_data))
    # a 2d array of the Newton basis evaluated on data
    # axis 0 = data value, axis 1 = iteration
    newton_basis = np.zeros((num_data, max_iterations))
    # the power function evaluated on data at each iteration
    # axis 0 = data value, axis 1 = iteration
    power_fct = np.zeros((num_data, max_iterations))
    # max power function in each iteration
    max_power_fct = np.zeros(max_iterations)
    # number of iterations
    num_iterations = max_iterations
    # variable to print training status
    fifty_iterations = 1

    output_dim = None
    transition_matrix = None
    newton_coeff = None
    residual = None
    max_residual = None
    rkhs_norm_surrogate = None
    surrogate = None
    kernel_coeff = None

    if data_dependent:
        # dimension of output space of f
        output_dim = f.shape[1]
        # basis transition matrix
        transition_matrix = np.zeros((max_iterations, max_iterations))
        # coefficients wrt the Newton basis
        # axis 0 = iteration, axis 1 = component
        newton_coeff = np.zeros((max_iterations, output_dim))
        # residual evaluated on data at each iteration
        # axis 0 = iteration, axis 1 = data value, axis 2 = component
        residual = np.zeros((max_iterations, num_data, output_dim))
        # 2-norm of the max residual in each iteration
        max_residual = np.zeros(max_iterations)
        # 2-norm of the mean residual in each iteration
        mean_residual = np.zeros(max_iterations)
        if f_is_rkhs:
            # interpolation error in rkhs norm ^2 at each iteration
            rkhs_norm_surrogate = np.zeros(max_iterations)

    # preparing output
    results = {}

    # Training
    print("Started training...")
    for k in range(0, max_iterations):
        # print training status
        if k > 50 * fifty_iterations:
            print("Selected more than", 50*fifty_iterations, "points...")
            fifty_iterations += 1

        # point selection in this iteration
        if k > 0:
```

```
113             selection_index = np.argmax(power_fct[notselected,k-1])
114         else:
115             selection_index = np.argmax(kernel_matrix_diagonal)
116         selected.append(notselected[selection_index])
117 
118         # computing the kth basis function
119         if k > 0:
120             a = newton_basis[notselected, 0:k].T
121             b = newton_basis[selected[k], 0:k]
122             kernel_values = kernel_matrix_vectorized(notselected, selected[k]).
                    ravel()
123             newton_basis[notselected, k] =  kernel_values - b @ a
124             newton_basis[notselected, k] /= power_fct[selected[k], k-1]
125         else:
126             kernel_values = kernel_matrix_vectorized(notselected, selected[k]).
                    ravel()
127             power = np.sqrt(kernel_matrix_diagonal[k])
128             newton_basis[notselected, k] = kernel_values / power
129 
130         # updating the power function
131         if k > 0:
132             power_squared = power_fct[notselected, k-1]**2
133         else:
134             power_squared = kernel_matrix_diagonal[notselected]
135         basis_squared = newton_basis[notselected, k]**2
136         power_fct[notselected, k] = np.sqrt(np.abs(power_squared - basis_squared)
                )
137         max_power_fct[k] = np.max(power_fct[:,k])
138 
139         if data_dependent:
140             # computing the kth coefficient wrt the Newton basis
141             if k > 0:
142                 r = residual[k-1, selected[k], :]
143                 p = power_fct[selected[k], k-1]
144                 newton_coeff[k, :] = r / p
145             else:
146                 power = np.sqrt(kernel_matrix(selected[k],selected[k]))
147                 newton_coeff[k, :] = f[selected[k], :] / power
148 
149             # updating the residual
150             if k > 0:
151                 r = residual[k-1, :, :]
152             else:
153                 r = f
154             cv = np.outer(newton_basis[:, k], newton_coeff[k, :])
155             residual[k, notselected, :] = r[notselected] - cv[notselected]
156             mean_residual[k] = np.mean(np.linalg.norm(residual[k,:,:], axis=1))
157             max_residual[k] = np.max(np.linalg.norm(residual[k,:,:], axis=1))
158 
159             # updating the transition matrix
160             t = transition_matrix[0:k, 0:k]
161             n = newton_basis[selected[k], 0:k].T
162             transition_matrix[0:k, k] = - t @ n
163             transition_matrix[k, k] = 1
164             if k > 0:
165                 transition_matrix[:, k] /= power_fct[selected[k], k-1]
166             else:
167                 transition_matrix[:, k] /= kernel_matrix(selected[0], selected
                        [0])
168 
169             # compute rkhs norm^2 of surrogate
170             if f_is_rkhs:
171                 rkhs_norm_surrogate[k] = np.sum(newton_coeff[0:k+1, :]**2)
172 
173         notselected.pop(selection_index)
174 
```

```
        # break if tolerance is met
        if max_power_fct[k] <= p_tol or max_residual[k] <= r_tol:
            # save expansion size
            num_iterations = k+1
            # cutting of not needed space for more iterations
            newton_basis = newton_basis[:, 0:num_iterations]
            power_fct = power_fct[:, 0:num_iterations]
            max_power_fct = max_power_fct[0:num_iterations]
            if data_dependent:
                newton_coeff = newton_coeff[0:num_iterations, :]
                residual = residual[0:num_iterations,:,:]
                mean_residual = mean_residual[0:num_iterations]
                max_residual = max_residual[0:num_iterations]
                transition_matrix = transition_matrix[0:num_iterations, 0:
                    num_iterations]
                if f_is_rkhs:
                    rkhs_norm_surrogate = rkhs_norm_surrogate[0:num_iterations]
            break

    # saving training Results
    results['selected'] = selected
    results['num_iterations'] = num_iterations
    results['max_power_fct'] = max_power_fct
    if data_dependent:
        # resulting approximation of data
        surrogate = newton_basis @ newton_coeff
        results['surrogate'] = surrogate
        # computing the coefficients wrt the kernel basis
        kernel_coeff = transition_matrix @ newton_coeff
        results['kernel_coeff'] = kernel_coeff
        results['residual'] = residual
        results['mean_residual'] = mean_residual
        results['max_residual'] = max_residual
        if f_is_rkhs:
            rkhs_error = norm_squarred - rkhs_norm_surrogate # squarred error
            results['rkhs_error'] = rkhs_error

    # print training results
    print("Completed Training.\n")
    print("Training results:\n")
    print("number of training data sites:", num_data)
    print("number of selected data sites/expansion size:", num_iterations)
    print("max of power function on training data:", max_power_fct[num_iterations
        -1])
    if data_dependent:
        print("mean residual on training data:", mean_residual[num_iterations-1])
        print("max residual on training data:", max_residual[num_iterations-1])
    if f_is_rkhs:
        print("RKHS error:", rkhs_error[num_iterations-1])

    return results
```

Listing B.1: pgreedy.py

B.2. Implemented Convlutional Neural Network according to LeNet-5 Architecture

```
import tensorflow as tf
import keras
from keras.datasets import mnist
from keras.models import Sequential
```

```
5  from keras.layers import Dense, Dropout, Flatten
6  from keras.layers import Conv2D, MaxPooling2D
7  from keras import backend as K
8  import numpy as np
9  import math
10
11 '''
12 Convolutional Neural Network by LeNet-5 Architecture
13
14 This script trains a convoltional neural network with the MNIST data base.
15 The architecture is according to LeNet-5. It trains the network, prints its
16 training results and saves the resulting feature vectors to output.npy
17 '''
18
19 def mnist_features(num_data_sites_train = None, num_data_sites_test = None):
20     # training parameters
21     batch_size = 128
22     num_classes = 10
23     epochs = 3
24
25     # input image dimensions
26     img_rows, img_cols = 28, 28
27
28     # data, split into train and test sets
29     (x_train, labels_train), (x_test, labels_test) = mnist.load_data()
30     x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
31     x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
32     input_shape = (img_rows, img_cols, 1)
33     x_train = x_train.astype('float32')
34     x_test = x_test.astype('float32')
35     x_train /= 255
36     x_test /= 255
37
38     # convert class vectors to binary class matrices
39     y_train = keras.utils.to_categorical(labels_train, num_classes)
40     y_test = keras.utils.to_categorical(labels_test, num_classes)
41
42     # sampling
43     if num_data_sites_train is not None:
44         len_train = len(x_train)
45         random_train_indeces = np.random.randint(len_train, size =
               num_data_sites_train)
46         x_train = x_train[random_train_indeces, :]
47         y_train = y_train[random_train_indeces, :]
48         labels_train = labels_train[random_train_indeces]
49     if num_data_sites_test is not None:
50         len_test = len(x_test)
51         random_test_indeces = np.random.randint(len_test, size =
               num_data_sites_test)
52         x_test = x_test[random_test_indeces, :]
53         y_test = y_test[random_test_indeces, :]
54         labels_test = labels_test[random_test_indeces]
55
56     # model
57     model = Sequential()
58     conv1 = Conv2D(6,
59                    kernel_size=(5, 5),
60                    activation='relu',
61                    padding = 'same',
62                    input_shape=input_shape)
63     model.add(conv1)
64     pool1 = MaxPooling2D(pool_size=(2, 2))
65     model.add(pool1)
66     conv2 = Conv2D(16, kernel_size=(5, 5), padding = 'valid',
67                      activation='relu')
68     model.add(conv2)
```

```
    pool2 = MaxPooling2D(pool_size=(2, 2))
    model.add(pool2)
    conv3 = Conv2D(120, kernel_size=(5, 5), padding = 'valid',
                     activation='relu')
    model.add(conv3)
    flat = Flatten()
    model.add(flat)
    model.add(Dense(84, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

    # training
    model.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adadelta(),
                  metrics=['accuracy'])
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              verbose=1,
              validation_data=(x_test, y_test))
    score = model.evaluate(x_test, y_test, verbose=0)
    print('Lenet Test Loss:', score[0])
    print('Lenet Test Accuracy:', score[1])

    # save feature vectors in output
    input = model.input
    outputs = [layer.output for layer in model.layers]
    functor = K.function([input, K.learning_phase()], outputs )
    layer_outs_train = functor([x_train, 1.])
    layer_outs_test = functor([x_test, 1.])
    features_train = layer_outs_train[5]
    features_test = layer_outs_test[5]
    output = [features_train, features_test, y_train, y_test, labels_train,
        labels_test]

    return output

if __name__ == "__main__":
    print("Saving output to output.npy...")
    np.save("output", mnist_features())
```

Listing B.2: mnistlenet.py

Hiermit versichere ich, dass die vorliegende Arbeit mit dem Titel *Kernel Interpolation: Theory and Applications* selbstständig verfasst worden ist, dass keine anderen Quellen und Hilfsmittel als die angegebenen benutzt worden sind und dass die Stellen der Arbeit, die anderen Werken – auch elektronischen Medien – dem Wortlaut oder Sinn nach entnommenen wurden, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht worden sind

Münster, December 16, 2018

(Celien Bosma)