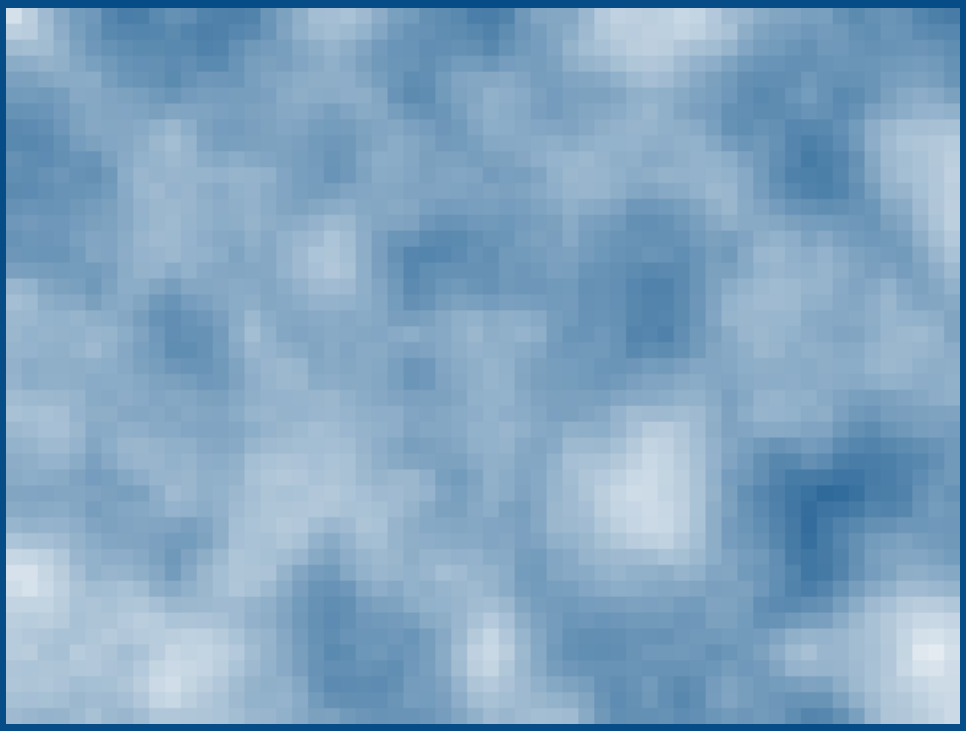
Lecture notes on Scientific Computing 2

Written by
Manuel Hinz
mh@mssh.dev or s6mlhinz@uni-bonn.de

Lecturer
Prof. Dr. Jochen Garcke
garcke@ins.uni-bonn.de



University of Bonn
Summer semester 2024
Last update: April 9, 2024

Contents

Chapter 0:
Manuel's notes and introduction

Warning

These are unofficial lecture notes written by a student. They are messy, will almost surely contain errors, typos and misunderstandings and may not be kept up to date! I do however try my best and use these notes to prepare for my exams. Feel free to email me any corrections to mh@mssh.dev or s6mlhinz@uni-bonn.de.
Happy learning!

General Information

- Ecampus: Ecampus link
- Basis: Basis link
- Website: https://ins.uni-bonn.de/teachings/ss-2024-440-v3e2-wissenschaftlich/
- Time slot(s): Tuesday 10-12 and Thursday 08-10
- Exams: Oral, unless more than 50 people take the exam
- Deadlines: tbd
- Two topics:
  - Kernel based methods for function approximation
  - Nonlinear dimensionality reduction / manifold learning / latent space embeddings
- Official lecture notes for most of the lectures
- Exercises are a mix of theory (proofs, (counter-)examples) and programming tasks

Start of lecture 01 (09.04.23)

0.1 Overview

We begin with a quick overview of the two parts of the lecture:

0.1.1 Function approximation / Interpolation

Consider $x_i \in \mathbb{R}^d, \hat{f}_i \in \mathbb{R}$:

$$\{(x_i, \hat{f}_i)\}_{i=1}^N.$$

Aim: Find a "good" function f such that

$$f(x_i) = \hat{f}_i, \ i = 1, \dots, N$$

Lagrangian Interpolation does not work great for a lot of points and higher dimensions

To compute f, we can make use of a discrete representation of f using **Ansatzfunctions** $\{b_j\}_{j=1}^N$:

$$f(x) = \sum_{j=1}^{N} c_j b_j(x).$$

Here we assume the same number of data and functions b_j.
For interpolation, we can solve this via:

$$BC = \hat{F}$$

Kernel functions that are centered at the locations x_j turn out to be a good choice:

$$b_j(x) = k(x_j, x)$$

which gives

$$f(x) = \sum_{j=1}^{N} c_j k(x_j, x).$$

We will also consider approximation instead of interpolation

$$f(x_i) \approx \hat{f}_i.$$

This is in particular relevant in machine learning, where one usually assumes, and actually has noise and measurement errors in the given data.
Example: Asses credit risk
Example: Chemistry / energy of molecules. This needs a kernel on graphs
Example: Time series. This needs a kernel on time series

Careful not to discriminate, credit risk should be independent of neighbourhood for example!

Remark. *We will also see that kernels relate to similarity measures and therefore to distances (dissimilarity).*

Topics in part 1:

- What are kernels and their properties
- Reproducing Kernel Hilbert spaces as the function space in which we are working
- Function interpolation and their approximation properties
- Generalized interpolation for solving partial differential equations
- Kernel methods for prediction in machine learning, representer theorem and regularization
- Gaussian Process Regression and Support Vector Machines

0.1.2 Dimensionality reduction

Distances and similarities are a key aspect of the second topic of the course:

Dimensionality reduction for high-dimensional data

The key idea is to find a "good" low dimensional representation (called embedding), such that chosen properties in high dimensions are approximately preserved.

- Linear dimensionality reduction (numerical linear algebra)
- Nonlinear dimensionality reduction (numerical linear algebra with Non-euclidean geometry)
- Dimensionality reduction with neural networks and other nonlinear function representations

Chapter 1:
Kernel based methods

1.1 Kernels

Definition (Gaussian kernel)**.** *The* ***gaussian kernel*** *is a prime example of a kernel:*

$$k(x,y) \coloneqq \exp\left(-\alpha\|x-y\|_2^2\right) = \phi(\|x-y\|_2)$$

for all $x, y \in \mathbb{R}^d$ *where* α *is a scaling parameter.*

Definition 1.1. *Let* Ω *be an arbitrary nonempty set. A function* $k : \Omega \times \Omega \to \mathbb{R}$ *is called* ***kernel*** *on* Ω*. We call* k *a* ***symmetric kernel*** *if*

$$k(x,y) = k(y,x)$$

for all $x, y \in \Omega$*.*

Definition 1.2. *A function* $\Phi : \mathbb{R}^d \to \mathbb{R}$ *is said to be* ***radial*** *if there exists a function* $\phi : [0,\infty] \to \mathbb{R}$ *such that*

$$\Phi(x) = \phi(\|x\|_2)$$

for all $x \in \mathbb{R}^d$*. Such a function is traditionally called a* ***radial basis function (rbf)****.*

1.1.1 Examples

Example ((Inverse) multiquadratics)**.** ***Multiquadratics*** *are of the form*

$$\phi(r) = (1+\alpha r^2)^\beta$$

for positive β*, while* ***inverse multiquadratics*** *have a* $\beta < 0$*.*

Example (Polyharmonic kernels)**.** ***Polyharmonic kernels*** *are of the form*

$$\phi(r) = r^\beta \log(|r|)$$

where $\beta \in 2\mathbb{Z}$*.*
The special case $\beta = 2$ *is the so-called* ***thin-plate spline****. It relates to the partial differential equation that describes the bending of thin plates.*

While the previous examples were monotone kernels (as a function of r*), these are not!*

Example (Wendland's kernels)**.** ***Wendland's kernels*** *are of the form*

$$\phi_{a,1} \coloneqq (1-r)_+^{(a+1)}(1+(a+1)r)$$

with the ***cut-off function***

$$(x)_+ \coloneqq \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Remark. *There are also non radial kernels:*
Translation-invariant *or* ***stationary*** *kernels are functions of differences:*

$$k(x, y) = \Phi(x - y).$$

For periodic setups, we have the ***Dirichlet kernel*** *as an example:*

$$D(\phi) := \frac{1}{2} + \sum_{j=1}^{N} \cos(j\varphi) = \frac{\sin\left(\left(n + \frac{1}{2}\right)\phi\right)}{2\sin\left(\frac{\phi}{2}\right)}.$$

This is applied to differences $\phi = \alpha - \beta$ of angles or 2π-periodic arguments and is an important tool for Fourier series theory.
There are so called ***zonal kernels****, for working on a sphere, where the kernel can be represented as a function of an angle. An example are functions of inner products, such as*

$$k(x, y) = \exp(x^\intercal y).$$

Remember, $x^\intercal y$ is the (scaled) cosine of the angle between the two vectors.

Remark. *We wil see that a kernel k on Ω defines a function $k(x, \cdot)$ for all fixed $x \in \Omega$. The space*

$$\mathcal{K}_0 := span\{k(x, \cdot) \mid x \in \Omega\}$$

can for example be used as a so called trial space in meshless methods for solving partial differential equations.

Remark. *Kernels can always be restricted to subsets without losing essential properties. This easily allows kernels on embedded manifolds, e.g. the sphere.*

Remark. *Most of this works for complex kernels too.*

1.1.2 Kernels in machine learning

In machine learning the data $x \in \Omega$ can be quite diverse and without (much) structure on first glance. For example consider images, text documents, customers, graphs, ...
Here, one views the kernel as a **similarity measure**, i.e.

$$k : \Omega \times \Omega \to \mathbb{R}$$

return a number $k(x, y)$ describing the similarity of two patterns x and y.
To work with general data, we first need to represent it in a Hilbert space $\mathcal{F}$, the so-called **feature space**. One considers the (application dependent) **feature map**

$$\Phi : \Omega \to \mathcal{F}.$$

The map describes each $x \in \Omega$ by a collection of **features** which are characteristic for a x and capture the essentials of elements of Ω. Since we are now in $\mathcal{F}$ we can work with linear techniques. In particular we can use the scalar product in $\mathcal{F}$ of two elements of Ω represented by their features:

$$\langle \Phi(x), \Phi(y) \rangle_{\mathcal{F}} =: k(x, y)$$

and define a kernel that way.

In $\mathbb{R}^d$, we can work with the standard scalar product

Reminder: A Hilbert space is a complete vector space with a scalar product

Remark. *Given a kernel, neither the feature map nor the feature space are unique, as the following example shows:*

Example. *Let $\Omega = \mathbb{R}, k(x, y) = x \cdot y$. A feature map, with feature space $\mathcal{F} = \mathbb{R}$ is given by the identity map.*
But,the map $\Phi : \Omega \to \mathbb{R}^2$ defined by

$$\Phi(x) := (x/\sqrt{2}, x/\sqrt{2})$$

is also a feature map given the same k!

Such a construction can be made for any arbitrary kernel, therefore every kernel has many different feature spaces

The following two examples show how one can handle non-euclidean origin spaces:

Example (Kernels on a set of documents)**.** *Consider a collection of documents. We represent each document as a* ***bag of words*** *and describe a bag as a vector in a space in which each dimension is associated with a term from the set of words, i.e. the dictionary. The feature map is*

that is a set of frequencies of (chosen) words

$$\Phi(t) \coloneqq (wf(w_1,t), wf(w_2,t), \dots, wf(w_d,t)) \in \mathbb{R}^d$$

where $wf(w_i,t)$ is the frequency of word w_i in document t.
A simple kernel is the vector space kernel

$$k(t_1,t_2) = \langle \Phi(t_1), \Phi(t_2) \rangle = \sum_{j=1}^{d} wf(w_j,t_1) wf(w_j,t_2).$$

Natural extensions to this kernel take e.g. word order, relevance or semantics into account, which can be achieved by using matrices in the scalar product:

$$k(t_1,t_2) = \langle S\Phi(t_1), S\Phi(t_2) \rangle = \Phi^\intercal(t_1) S^\intercal S \Phi(t_2.)$$

Example (Graph kernels)**.** *Another non-euclidean data object are graphs, where the class of* ***random walk kernels*** *can be defined. These are based on the idea that given a pair of graphs, one performs random walks on both and counts the number of matching walks. With $\tilde{A}_\times$ the adjacency matrix of the* ***direct product graph*** *of the two involved graphs, one defines:*

$$k(G,H) \coloneqq \sum_{j=1}^{N_G} \sum_{k=1}^{N_H} \sum_{l=1}^{\infty} \lambda_l [\tilde{A}_\times^l]_{j,k}.$$

More generally, one can define a ***random walk graph kernel*** *k as*

$$k(G,H) \coloneqq \sum_{k=0}^{\infty} \lambda_k q_\times^T W_\times^k p_\times,$$

where $W_\times$ is the ***weight matrix*** *of the direct product graph, $q_\times^T$ is the* ***stopping probability*** *on the direct product graph, and $p_\times$ is the initial product distribution on the direct product graph.*

1.1.3 Mercer kernels

More generally, one can consider kernels of the **Hilbert-Schmidt** or **Mercer** form

$$k(x,y) = \sum_{i \in I} \lambda_i \varphi_i(x) \varphi_i(y),$$

with certain functions $\varphi_i : \Omega \to \mathbb{R}$, certain positive **weights** λ_i and an index set I such that the following **summability condition** holds for all $x \in \Omega$:

$$k(x,y) = \sum_{i \in I} \lambda_i \varphi_i(x)^2 < \infty \tag{1}$$

Remark. *Such kernels arise in machine learning if the functions φ_i each describe a feature of x and the feature space is the weighted l_2-space of sequences with indices in I:*

$$l_{2,I,\lambda} \coloneqq \left\{ \{\xi_i\}_{i \in I} : \sum_{i \in I} \lambda_i \xi_i^2 < \infty \right\}.$$

This expansion also occurs when kernels generating positive operators are expanded into eigenfunctions on Ω. Such kernels can be views as arising from generalized convolutions. Generally kernels have three major application fields:

- Convolutions

- Trial spaces
- Covariances

We are mainly concerned with the last two.