Geometric learning for symmetric positive definite matrices

Oliver Potocki

Born 3rd May 1994 in Herdecke

September 28, 2023

Master's Thesis Mathematics

Advisor: Prof. Dr. Michael Griebel

Second Advisor: Prof. Dr. Jochen Garcke

Institut für Numerische Simulation

Mathematisch-Naturwissenschaftliche Fakultät der

Rheinischen Friedrich-Wilhelms-Universität Bonn

Abstract

Machine learning, while transformative in various domains, faces challenges when dealing with data in non-Euclidean spaces. Symmetric positive definite (SPD) matrices, frequently encountered in fields like computer vision and medical imaging, exemplify this challenge. These matrices, often representing regularized covariance matrices, are more naturally modeled using Riemannian manifolds rather than vectors in Euclidean space. This thesis delves into three Riemannian metrics for modeling SPD matrices: the affine-invariant Riemannian metric, the Log-Euclidean metric, and the Bures-Wasserstein metric. We explore their derivations, computational aspects, and invariances. While many learning methodologies have adopted the Log-Euclidean framework due to its vector space structure, this work places focus on the investigation of the Bures-Wasserstein kernel in image classification and kernel ridge regression contexts. Through a series of numerical experiments, we assess the properties and performance of these geometries in machine learning tasks, offering insights into their potential applications and limitations. Our findings are part of the broader effort to integrate geometry with machine learning, ensuring that the intrinsic geometry of the data is respected. Consequently, it became clear that the Bures-Wasserstein approach is a suitable alternative for learning on SPD matrices.

Acknowledgements

I thank Professor Griebel for his counsel and advice throughout the writing of this thesis. Through his mentorship, I gained a deeper understanding of mathematics and the role of a mathematical researcher. I also express my gratitude to Professor Garcke for evaluating my thesis. To my friend Konstantin, I thank you for reading the final draft of this thesis and for your helpful suggestions. Finally, I would like to express my heartfelt thankfulness to my family for their love and support throughout the year.

Contents

List of Figures

List of Tables

Notation

$\mathbb{R}$	Real numbers
$\mathbb{R}^d$	d-dimensional real vector space
$\mathbb{R}^{d\times d}$	$d \times d$ real matrix vector space
$\mathrm{Sym}(\mathbb{R}^d)$	Vector space of $d \times d$ symmetric matrices over $\mathbb{R}^d$
$\mathrm{Sym}_+(\mathbb{R}^d)$	Set of $d \times d$ positive semi-definite matrices over $\mathbb{R}^d$
$\mathrm{Sym}_{++}(\mathbb{R}^d)$	Set of $d \times d$ positive definite matrices over $\mathbb{R}^d$
x	Vectors are written in small letters
A	Matrices are written in capital letters
$[N]$	denotes the set $\{1, 2, \ldots, N\}$.
$\mathcal{M}$	Typically represents a manifold
$\mathcal{X}$	Domain from which training data points are drawn
$\mathcal{Y}$	Set of target values in a supervised learning problem
$\mathcal{R}$	Generalization error in a learning problem
$\hat{\mathcal{R}}$	Empirical generalization error based on training data
J	An objective functional in optimization problems
$\mathrm{GL}(\mathbb{R}^d)$	General linear group of invertible dxd matrices over $\mathbb{R}^d$
$\mathrm{O}(\mathbb{R}^d)$	Orthogonal group of dxd orthogonal matrices over $\mathbb{R}^d$
$\mathcal{L}(H, G)$	Set of bounded linear operators from Hilbert space H to G
$\mathcal{L}(H)$	Shorthand for $\mathcal{L}(H, H)$

Chapter 1

Introduction

Machine learning is becoming ubiquitous as a paradigm for computing in science and industry. It delivered stunning results in areas as diverse as image generation [Ramesh et al., 2022], protein folding prediction [Jumper et al., 2021], or playing games as complex as Go [Silver et al., 2016].

However, machine learning still struggles with treating data lying in non-Euclidean spaces. Only recently, with the advent of techniques such as such statistics for random variables taking values on manifolds [Pennec, 2006] or algorithms like Graph Neural Networks [Kipf and Welling, 2017] have learning techniques begun to respect the geometric structure of data intrinsically. For data with manifold-valued outputs, consistent frameworks are rare. They either work ad hoc or rely on restrictive assumptions on the output manifold. One reason is that for tasks such as regression or regularization in different norms or metrics, typical assumptions such as statistical consistency of estimators or even the existence of expectations do not necessarily hold and must be carefully adapted.

Symmetric positive definite (SPD) matrices are an increasingly important data type. In many cases, SPD matrices appear as regularized covariance matrices describing the covariance of data points or the distribution of features in a data instance. In covariance region descriptors [Tuzel et al., 2006] they encode the covariance between features such as intensity, RGB color values, or color gradient, whereas in Diffusion Tensor Imaging [Bihan et al., 2001] they represent the local diffusivity of water molecules in tissue.

Treating SPD data as a vector in a Euclidean space is possible but ignores any specific structure the SPD matrices have. Often, the most suitable distance to compare or interpolate SPD matrices is given by a Riemannian metric on a manifold of SPD matrices. In contrast, the Euclidean distance might lead to artifacts such as the swelling effect in white matter tractography.

The field of geometric statistics [Pennec, 2006] arose in order to extend statistical methodology to the case of Riemannian manifolds. Concurrently, there has also been much development on software packages for algorithms that inherently work in a non-Euclidean setting, both for computational statistics

[Miolane et al., 2020a] and optimization [Boumal, 2023].

In recent years, the desire to model data more accurately for learning and optimization problems has driven research in abstract mathematics that combines algebraic and geometric ideas. The affine-invariant Riemannian metric [Pennec et al., 2004] was defined to provide a framework for geometric statistics on symmetric positive definite matrices. By requiring invariance to the action of the general linear group, the Frobenius scalar product on the tangent space of SPD matrices becomes a Riemannian metric with strong regularity properties capturing the canonical differential structure of SPD matrices. This allowed the definition of inherent Riemannian algorithms that respect the geometry and show strong results on tasks such as interpolation or regularization. However, the high computational cost of this metric and the lack of closed form expressions for objects such as the mean motivated the search for other geometric structures. The Log-Euclidean metric [Arsigny et al., 2006] was found while searching for a Lie group structure on the SPD matrices. Moreover, it defines not only such a group but also a vector space, making it possible to transfer many of the usual expressions and methods from Euclidean space to the logarithmic domain. Over time, many different geometries have been defined, most recently the Bures-Wasserstein metric [Bhatia et al., 2019], which is the Wasserstein-2 distance of multivariate normal distributions expressed as a Riemannian metric on the manifold of their covariance matrices.

This thesis will provide a comprehensive overview of how the different geometries of SPD matrices can be effectively utilized in machine learning tasks. We will reflect on the different implications of using one particular metric over another by looking at the invariances defined by the respective geodesic distances and by considering the computational aspects of the different constructs.

Several works have extended both kernel methods [Jayasumana et al., 2015, Feragen et al., 2015] and neural networks [Huang and Gool, 2017] to SPD matrices. In most cases, learning methodologies adapt the Log-Euclidean framework as its vector space structure provides a notion of linearity, making it feasible to generalize machine learning to the manifold case. In contrast, the Bures-Wasserstein geometry has been explored little for purposes of learning. To the best of our knowledge, we will provide the first investigation of the Bures-Wasserstein kernel in both image classification and kernel ridge regression. The latter will also touch on the question of how the Log-Euclidean and Bures-Wasserstein distances compare as approximations of the affine-invariant Riemannian manifold.

The rest of this thesis is structured as follows, first in Chapter 2 we review the basics of differential geometry and statistical learning. A special focus here will be on the interplay between geometry and statistics, where we will see that the change of the underlying space has profound implications for the central concepts of probability.

Following that, we will introduce several different geometries on the SPD matrices in Chapter 3. The affine-invariant Riemannian metric, the Log-Euclidean metric, and the Bures-Wasserstein metric are three ways to define geodesic distances on the SPD matrices. We will look at their invariance properties and corresponding costs to assess their suitability for learning and optimization tasks.

In Chapter 4, we introduce the theory of reproducing kernel Hilbert spaces and kernel methods using implicit embeddings. We will investigate the possibility of defining positive definite kernels on sets of SPD matrices using Gaussian kernels with different geodesic distances. The theory of vector-valued reproducing kernel Hilbert spaces will lead to an exposition of the kernel sum-of-squares model for functions taking values in the cone of positive semi-definite matrices.

Neural networks are treated in Chapter 5. After a brief review of the theory of neural networks in the Euclidean case, we will see how the SPDnet architecture enables us to learn about SPD data while inherently respecting its geometry. At the end of this chapter we will give a short outlook on the still open question of defining neural networks for functions taking values on the manifold.

The differences in the properties of the different geometries will then be investigated by means of several numerical experiments in Chapter 6. We will perform support vector-based classification on sets of covariance descriptors to see how different kernels provide embeddings in feature spaces, visualize the separation of classes, and compare their accuracy to each other and to an instance of SPDnet. Subsequent kernel ridge regression of the geodesically linear logdet map provides insight into the ability of the different metrics to approximate the affine-invariant metric. Interpolation of the affine-invariant geodesic with the kernel sum-of-squares model tests the ability of vector-valued kernel methods to perform regression on the cone of SPD matrices.

We conclude with some final remarks and reflections in Chapter 7.

Chapter 2

Statistical Learning

In this chapter, we will introduce the basic notions and the mathematical background of statistical learning and optimization theory, with special consideration to the manifold case, that will be used throughout this thesis. We will begin by setting up general notions of differential geometry and statistics for Riemannian manifolds following [Guigui et al., 2023]. The second section will be devoted to empirical risk minimization with a focus on supervised learning. For this exposition, we will take guidance from the text of [Mohri et al., 2018].

2.1 Riemannian manifolds

A manifold is a topological space that is locally Euclidean but might possess a global non-linear structure. Most numerical procedures will be local in nature, with possible corrections for the non-Euclidean nature. In the case of learning algorithms, most are some form of local weighted averaging. We restrict ourselves to manifolds that are embedded in a Euclidean space and have a smooth transition map between local neighborhoods. Formally, we define a differentiable manifold by

Definition 2.1.1 (Differentiable manifold)**.** We call a nonempty set $\mathcal{M} \subset \mathbb{R}^d$ a m-dimensional, *differentiable manifold* if for every point $p \in \mathcal{M}$ there are two open subsets $V \subset \mathbb{R}^m$ and $U \subset \mathbb{R}^d$, such that there exists a smooth function $f\colon V \to \mathbb{R}^d$ with $f(0) = p$ and $f(V) = U \cap \mathcal{M}$. Moreover, if $f_1\colon V_1 \to U_1$ and $f_2\colon V_2 \to U_2$ are two such functions with $f_1(V_1) \cap f_2(V_2) \neq \emptyset$, then the transition map $f_2^{-1} \circ f_1\colon f_1^{-1}(f_1(V_1) \cap f_2(V_2)) \to f_2^{-1}(f_1(V_1) \cap f_2(V_2))$ is also smooth.

Another very important notion is that of a tangent space. Intuitively a tangent space is like a hyperplane in the ambient space that touches the manifold on a single point. For local neighborhoods around that point a tangent space is often a good approximation for the manifold. Most importantly a tangent space is a Euclidean vector space providing us with a notion of linearity and a scalar product that will prove very useful for numerical computation.

Definition 2.1.2. Let $\mathcal{M}$ be a manifold in $\mathbb{R}^d$ of dimension m, and $p \in \mathcal{M}$. A vector $v \in \mathbb{R}^d$ is tangent to $\mathcal{M}$ at p if there exists an open interval I centered around 0, and a curve $\gamma\colon I \to \mathcal{M}$ such that $\gamma(0) = p$ and $\dot{\gamma}(0) = v$. We call the set of tangent vectors $T_p\mathcal{M}$ the *tangent space* at p.

Given a point p on a manifold $\mathcal{M}$ and the tangent space $T_p\mathcal{M}$, we define a scalar product on $T_p\mathcal{M}$ as a positive definite, symmetric, bilinear function $\langle \cdot, \cdot \rangle_p \colon T_p\mathcal{M} \times T_p\mathcal{M} \colon \to \mathbb{R}$. A *Riemannian metric* is a collection of scalar products that varies smoothly in p.

We will call a manifold Riemannian if it is equipped with such a metric. As usual, the scalar product also induces a norm for tangent vectors $||x||_p = \sqrt{\langle x, x \rangle_p}$. Using this we can define a distance measure on the manifold. Taking a curve $\gamma\colon [a, b] \to \mathcal{M}$ and the tangent vector at each point of the curve $\dot{\gamma}(t)$ we define a variational functional

$$E(\gamma) = \frac{1}{2} \int_a^b ||\dot{\gamma}(t)||^2_{\gamma(t)} dt. \tag{2.1}$$

A *geodesic* is a curve that is a minimum of the functional 2.1. A geodesic has zero acceleration that is

$$\frac{d}{dt} ||\dot{\gamma}(t)||_{\gamma(t)} = 0. \tag{2.2}$$

With this property, geodesics can be seen as the generalization of straight lines onto manifolds. To build up the framework of geometric statistics as defined by [Pennec, 2006] we will, in the following assume all manifolds to be connected and geodesically complete, which means that for any start point, we can follow any geodesic indefinitely without hitting any boundaries or singularities. By the Hopf-Rinow theorem, this also implies that a length-minimizing geodesic between any two points on the manifold always exists.

For a geodesically complete manifold $\mathcal{M}$ we define the geodesic distance $d_g \colon \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ as the length of the length-minimizing geodesic joining the two points. Note that the geodesic between points need not be unique.

Many algorithms are defined by a computation on the tangent space of a point $T_p\mathcal{M}$, which is a Hilbert space with an inner product defined by the Riemannian metric. They are then mapped back to the manifold with the use of the *Riemannian exponential map.* The Riemannian exponential map $\mathrm{Exp}_p \colon \mathcal{U} \to \mathcal{M}$ is defined by

$$\mathrm{Exp}_p(y) = \gamma_y(1). \tag{2.3}$$

That is, we map $y \in T_p\mathcal{M}$ to the point $q \in \mathcal{M}$, such that for a geodesic with $\gamma(0) = p$ and $\dot{\gamma}(0) = y$ we have $\gamma(1) = q$. For a point $p \in \mathcal{M}$, we define the local injectivity radius by

$$r_{inj}(p) = \sup_{r>0} \{ \mathrm{Exp}|_{B_r(0)} \text{ is diffeomorpism} \} \tag{2.4}$$

for a ball $B_r(0) \subset T_p\mathcal{M}$. For a whole manifold, we define the injectivity radius $r_{inj}(\mathcal{M})$ as the infimum over the injectivity radii of all points on the manifold.

The inverse of the exponential map is the logarithm map which goes from the manifold to a tangent space

$$\mathrm{Log}_p \colon B_r(p) \to \mathcal{U} \subset T_p\mathcal{M} \tag{2.5}$$

for $r \geq r_{inj}(\mathcal{M})$.

In fact, there is a whole zoo of mathematical constructs that further specialize manifolds. Depending on our learning goal, it can be beneficial to use multiple perspectives on the same data set. Incorporating algebraic aspects in addition to differential geometry proves beneficial in the context of learning. Of particular interest is the *homogeneous space*. To define it, we first recall the definition of a group

Definition 2.1.3 (Group)**.** A *group* is a set G together with a binary operation $\cdot : G \times G \to G$ that satisfies the following properties:

1. *Closure*: For all $a, b \in G$, the result of $a \cdot b$ is also in G, i.e., $a \cdot b \in G$.
2. *Associativity*: For all $a, b, c \in G$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
3. *Identity element*: There exists an element $e \in G$, called the *identity element*, such that for all $a \in G$, $a \cdot e = e \cdot a = a$.
4. *Inverse element*: For each $a \in G$, there exists an element $a^{-1} \in G$, called the *inverse element* of a, such that $a \cdot a^{-1} = a^{-1} \cdot a = e$.

Groups can be thought of as an abstract formulation of the idea of symmetry. Symmetry, in particular, implies the redundancy of certain parts of the data, and incorporating it will often prove essential to make processing more powerful and efficient. Next, we define how a group interacts with a general set by the notion of group action

Definition 2.1.4 (Group Action)**.** Let G be a group, and let X be a set. A *group action* of G on X is a function $. : G \times X \to X$ such that for all $g, h \in G$ and $x \in X$, the following properties hold:

1. *Identity Action*: $e.x = x$, where e is the identity element of G.
2. *Compatibility*: $(g.h).x = g.(h.x)$ for all $g, h \in G$ and $x \in X$.

Using this, we can define a very regular kind of manifold incorporating additional regularity

Definition 2.1.5 (Homogeneous space)**.** We call a manifold $\mathcal{M}$ a *homogeneous space*, if there exists a group G such that its action on $\mathcal{M}$ is transitive, that is

$$\text{For all } x, y \in \mathcal{M}, \text{ there exists } g \in G \text{ such that } g.x = y. \tag{2.6}$$

A homogeneous space is a very natural place for learning, because we can detect local patterns which will look the same everywhere in the space. Furthermore, the transitive group defines a canonical way to move inside our data.

An important identity for understanding homogeneous spaces is the characterization as quotient space. If $\mathcal{M}$ is a homogeneous space with a group G acting on it we have that

$$\mathcal{M} \simeq G/H \tag{2.7}$$

where H is a subgroup of G called the stabilizer. $\mathcal{M}$ can therewith be seen as the set of orbits of H on G.

2.2 Empirical risk minimization on manifolds

In the setting of *supervised learning* we are given two sets $\mathcal{X}$ and $\mathcal{Y}$ and a *training set* of tuples $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, which we assume to be drawn independent and identically distributed from $\mathcal{X} \times \mathcal{Y}$ according to a measure μ. It should be noted that we assume that there is possible noise on the labels. The aim is to approximate a function $f \colon \mathcal{X} \to \mathcal{Y}$ such that

$$f(x) \approx y \tag{2.8}$$

The dissimilarity of the approximated output $f(x)$ to the true label y will be quantified by a *loss function* $l \colon \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, which in our case will be different distance measures on a manifold or metric space. However other measures such as divergences are also possible to express similarity. We refer to the regime in which we require exact correspondence at the training points as interpolation, while reserving the term approximation for the more general case.

A foundational task in machine learning is *classification*, where we are given some data drawn from $\mathcal{X}$ and want to put elements into certain predefined categories, that is $\mathcal{Y} = \{1, \cdots, N\}$. Examples might be photos that should be classified as either cats or dogs, or medical images where certain diseases might be detected. The case where $\mathcal{Y} = \{0, 1\}$ is called binary classification as opposed to multi class classification for more than two categories. Theory is often reduced to the former case as every classification problem can be seen as collection of binary problems where for each class we have to decide whether the data point belongs to the class or not.

If the response variable in the output space is - in some sense - continuously dependent on the input we speak of a *regression* problem. Typical examples are weather forecasting or the analysis of historical trends.

Statistical learning is interested in the error for the whole distribution and aims to find functions that do well on previously unseen data. We can then formulate the overall goal of supervised learning as the minimization of a risk function, often also called the *generalization error*

$$\min_{f \in \mathcal{Y}^{\mathcal{X}}} \mathcal{R}(f) = \mathbb{E}_\mu[l(f(x), y)] = \int_{\mathcal{X} \times \mathcal{Y}} l(f(x), y) d\mu(x, y) \tag{2.9}$$

If we take $\mathcal{Y}$ to be a Riemannian manifold with the squared geodesic distance $d^2_{\mathcal{Y}}$ as loss function, we can follow the factorization argument in [Steinke et al., 2010] to see that the *Bayes estimator*, the function achieving the lowest possible risk, is given pointwise by

$$f^*(x) = \underset{p \in \mathcal{Y}}{\operatorname{argmin}} \int_{\mathcal{Y}} d^2_{\mathcal{Y}}(p, y) d\mu_x(y) \tag{2.10}$$

Where $d\mu_x(y)$ is the conditional expectation of y given x. The global minimizer of this integral is called the *Karcher mean* [Karcher, 1977]. It is the generalization of the Euclidean mean in the sense that it is the point minimizing the overall variance. One thing to keep in mind is that in a manifold setting the existence or uniqueness of this expectation cannot be guaranteed. Indeed, the formulation of consistent stochastic frameworks in which common expressions such as mean or variance can be reasoned about is one of the main motivations for the geometries later presented in Chapter 3. Furthermore, we will not know the measure μ. In fact, statistical learning theory in the general case goes so far as to make no assumptions at all about this generative measure.

Searching for such a function in the class of all measurable functions $\mathcal{Y}^{\mathcal{X}}$ is generally infeasible. Therefore, we will restrict the class of functions we consider as solutions to some $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$, which we call the *hypothesis space* or *model class.* The appropriate choice of space for a type of problem is one of the core issues of learning. The *No Free Lunch theorem* tells us for every possible hypothesis class there exists a distribution that will give arbitrarily slow convergence to the minimal possible loss [Devroye et al., 1996]. So there is no class that will perform universally well on all problems, therefore, we will always have to be very careful how we constrain the overall function space.

Models can be parametric or non-parametric. A parametric function class is fully described by a set of values $\Theta \subset \mathbb{R}$, independent of sample size. An example of a parametric model class might be the affine functions for $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$:

$$\mathcal{F}_{\text{Aff}} = \left\{ f \colon \mathbb{R}^d \to \mathbb{R} \;\middle|\; f_\theta(x) = b + \langle w,\, x \rangle \right\} \tag{2.11}$$

Here $\theta = (w, b)$ are our parameters. w is typically referred to as weights and b as the bias. An example for an unparametric function class are functions described by positive definite kernels, see Chapter 4.

For the case where $\mathcal{Y}$ is a manifold defining function classes is more intricate. For functions going into vector spaces we define a respective function vector space by pointwise addition and scalar multiplication of the outputs, which is not possible in the manifold case. Indeed the set of manifold valued functions is often itself an infinite dimensional manifold complicating approximation. The recipe of finding a finite basis set from which to construct predictors breaks down. Frameworks that treat the general learning case here are rare, and are often based on restrictive assumptions such as quasi-uniformity of the data [Grohs, 2012], equivalence of the loss to a billinear product in a Hilbert space [Rudi et al., 2018] or restricted to the optimization over perfectly smooth functions [Steinke et al., 2010].

Another avenue is to only consider a local approximation, doing away with the global nonlinear structure or bound it by a correction term. By taking a compact subset it is possible to find an isometric embedding into the Euclidean vector space and use the dual structure there.

Moving on, another complication is that we will have to use our training data to make an empirical estimate of the true risk. The central limit theorem tells us that under mild conditions the distribution of the mean of repeated measurements of random variables tends towards a Gaussian distribution, motivating an assumption of centered normally distributed noise. Given now, that the squared Euclidean differences as a loss is equivalent to a maximum likelihood estimator of such a distribution, we get a very standardized form of empirical risk in the Euclidean case

$$\min_{f \in \mathcal{F}} \hat{\mathcal{R}}(f) = \frac{1}{N} \sum_{i=1}^{N} ||y_i - f(x_i)||_2^2. \tag{2.12}$$

This is not as straightforward in the manifold case. First of all there is no unambiguous generalization of the Gaussian distribution. In a statistical learning setting, one can follow [Fletcher, 2020] and define the normal distribution on a manifold by the probability density function (pdf)

$$\rho(y; p, \tau) = \frac{1}{C(p,\tau)} \operatorname{Exp}\left(-\frac{\tau}{2} d_{\mathcal{Y}}^2(p,y)\right) \tag{2.13}$$

with a normalization constant

$$C(p,\tau) = \int_{\mathcal{Y}} \operatorname{Exp}\left(-\frac{\tau}{2} d_{\mathcal{Y}}^2(p,y)\right) d\mu(y). \tag{2.14}$$

Assuming such Riemannian normal noise, using the empirical Karcher mean

$$\hat{f^*}(x) = \operatorname*{argmin}_{p \in \mathcal{Y}} \frac{\tau}{2} \sum_{i=1}^{N} \alpha_i(x) d_{\mathcal{Y}}^2(p, y_i) \tag{2.15}$$

we define a consistent estimator of the Bayes predictor under the condition that $\mathcal{Y}$ is a homogeneous space and suitable weight functions α_i. The homogeneity condition guarantees that the normalization constant does not depend on the location on manifold.

The minimal realizable risk is called the *Bayes risk*

$$\mathcal{R}^* = \inf_{f \in \mathcal{Y}^{\mathcal{X}}} \mathcal{R}(f). \tag{2.16}$$

It if often assumed to be strictly larger than zero reflecting fundamental noise that cannot be dissolved meaningfully. For $\hat{f} \in \operatorname{argmin}_{f \in \mathcal{F}} \hat{\mathcal{R}}(f)$ the minimum discrepancy risk can now be decomposed

$$\mathcal{R}(\hat{f}) - \mathcal{R}^* = \underbrace{\left(\mathcal{R}(\hat{f}) - \min_{f \in \mathcal{F}} \mathcal{R}(f)\right)}_{\text{estimation error}} + \underbrace{\left(\min_{f \in \mathcal{F}} \mathcal{R}(f) - \mathcal{R}^*\right)}_{\text{approximation error}}. \tag{2.17}$$

Here the approximation error is independent of the size of the training data and will generally decrease with increasing complexity of the hypothesis space. On the other hand, the estimation error decreases as the sample size grows, but worsens as the complexity of the search space rises. This conundrum is called the *bias-variance trade-off*. In practice the dominant paradigm to approach this is regularization. Here a large hypothesis class is chosen, while an additional penalty is added to the risk functional

$$J(f) = \mathcal{R}(f) + \lambda\Omega(f). \tag{2.18}$$

λ is a hyperparameter selected apriori and controls the strength of regularization. In many cases we will try to stir our solution in direction of a low-complexity solution, such as one of small norm. Defining Ω norm dependent has the additional benefit of convexifying our problem and possibly guaranteeing uniqueness of the solution. Defining a notion of smoothness for manifold valued functions is nontrivial and there are many possible approaches depending on the use case. In particular, the lack of vector space structure of our model class makes the definition of Sobolev spaces difficult.

By restricting our model class to smooth functions, as was done in [Steinke et al., 2010], we can use the norm of covariant derivatives of differentials to define a regularization functional. It is well known however that approximating with smooth functions can lead to non-smooth limits due to missing closure.

Nevertheless the most popular ansatz in practice is to combine the sum of geodesic square loss with the norm of Riemannian derivatives, in the simplest case the Riemannian gradient, to get the empirical objective functional

$$J(f) = \frac{1}{N}\sum_{i=1}^{N} d^2_{\mathcal{Y}}(f(x_i), y_i) + ||\nabla^\alpha f||^2. \tag{2.19}$$

for some multi-index α. In practice this is most often the Dirichlet energy or a harmonic or Hessian regularization.

2.3 Conclusion

In many applications, modeling data as being on a manifold has proven advantageous, giving rise to a rich field of research ever aiming to both provide a rigorous mathematical framework as well as practical methods. The interplay between abstract differential geometric and algebraic formalism with statistical learning pushes us to reexamine many assumptions we took for granted. Ideas central to statistics, such as the central limit theorem, no longer hold in full generality. Much as the introduction of non-Euclidean geometry provided the tools for new ways of thinking in physics, we believe that expanding the universe of function approximation can help us understand how learning works at a fundamental level.

Nevertheless, the way still seems long. As of now, we lack a consistent theory for statistical learning in even very regular and smooth structures. Further, there seem to be fundamental issues looming, as losing the notion of linearity might be deemed fatal for understanding the whole phenomenon. On the other hand, locality might suffice for most practical needs, and perhaps shifting paradigms to include entirely local structures will enrich our perspectives. There are also challenges on the numerical side, as algorithms in geometric statistics often scale only to smaller data sets and can suffer from numerical instability. Even a simple task like fitting a geodesic curve can be fraught with gradients that explode as noise that is small in tangent space becomes exponential when projected onto the manifold.

In the following chapters, we will explore this evolving domain of geometric learning on a specific example, the symmetric positive definite matrices. We will see how the choice of different geometric frameworks affects not only the computation, but also our understanding of the underlying data.

Chapter 3

Symmetric Positive Definite Matrices

In this chapter we will study different metrics on symmetric positive definite matrices, their geometric properties, and how to use them for learning and optimization. We will introduce three different Riemannian metrics. One is the affine-invariant Riemannian metric under which the SPD matrices become a Cartan-Hadamard manifold, a second is the Log-Euclidean metric which gives the SPD matrices a unique vector space structure. Both induce geodesically complete manifolds. Finally, we will also look at the Bures-Wasserstein metric under which the SPD matrices become a manifold with boundary.

A distance can be thought of as a similarity measure, and the geometry it induces as a constraint. We will therefore focus on the invariances inherent to the different manifolds. As we will see, imposing different symmetries on the distances between SPD matrices has strong computational implications. In particular, an invariance for a smaller group can facilitate optimization in the geometry characterized by that group action. It should be noted, however, that the geometries are more than just a means to an end; which geometry we choose will be determined by which criteria we consider SPD matrices to be similar. Thus, we will also dive into some connections and interpretations of the different metrics to understand what it is they encode.

A square matrix $A \in \mathbb{R}^{d \times d}$ is called *symmetric positive definite* (SPD) if it satisfies the following two conditions:

1. A is symmetric: $A = A^T$
2. A is positive definite: for all non-zero vectors $x \in \mathbb{R}^d$, $x^T A x > 0$

We will call the set of $d \times d$ SPD matrices $\mathrm{Sym}_{++}(\mathbb{R}^d)$. At times we will suppress the space for better readability. Note that the SPD matrices form an open, convex subset of the vector space of symmetric matrices

$$\mathrm{Sym}(\mathbb{R}^d) = \{M \in \mathbb{R}^{d \times d} \mid M = M^T\}. \tag{3.1}$$

Symmetric positive definite matrices are ubiquitous mathematical objects. They appear inducing bilinear forms or as optimality conditions and constraints in numerical problems. SPD matrices always have positive real eigenvalues, and a natural way to represent them is via their singular value decomposition (SVD). Let $A \in \mathrm{Sym}_{++}(\mathbb{R}^d)$, we can then present A via

$$A = W^T D W \tag{3.2}$$

where W is the matrix of eigenvectors of A and a member of the orthogonal group

$$\mathrm{O}(\mathbb{R}^d) = \{M \in \mathbb{R}^{d\times d} \mid M^T M = I\} \tag{3.3}$$

consisting of rotations and reflections. D is a diagonal matrix consisting of the eigenvalues of A. It is a common convention to order the eigenvalues by their size, with D_{11} being the largest and D_{dd} being the smallest eigenvalue.

Since we can write powers of matrices as $A^k = W^T D^k W$, we reduce the matrix exponential to the exponential function of the real numbers applied pointwise to each eigenvalue

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!} = W^T \mathrm{diag}(\exp(D_{ii}))W. \tag{3.4}$$

As the eigenvalues of SPD matrices are always positive the inverse of this function, the matrix logarithm, is well-defined everywhere as

$$\log(A) = W^T \mathrm{diag}(\log(D_{ii}))W \tag{3.5}$$

These functions form global diffeomorphisms between the space of symmetric positive definite matrices and the space of symmetric matrices.

We will look at symmetric positive definite matrices as geometric objects and find natural structures that facilitate learning. We will first investigate the defects that appear when treating SPD matrices as Euclidean vectors.

3.1 Euclidean metric

As mentioned $\mathrm{Sym}_{++}(\mathbb{R}^d)$ is a subset of the vector space of symmetric matrices $\mathrm{Sym}(\mathbb{R}^d)$, which in turn is a sub-vector space of the space of square matrices $\mathrm{Mat}(d) \simeq \mathbb{R}^{d\times d} \simeq \mathbb{R}^{d^2}$. The vector space $\mathrm{Sym}(\mathbb{R}^d)$ is of dimension $\frac{d(d+1)}{2}$ as symmetric matrices can be completely represented by their upper-triangonal entries. For $A, B \in \mathrm{Sym}(\mathbb{R}^d)$ the Euclidean inner product for this vector space is usually taken in form of the Frobenius inner product

$$\langle A, B\rangle_F = \mathrm{Tr}(A^T B) = \sum_{i,j=1}^{d} A_{ij} B_{ij}. \tag{3.6}$$

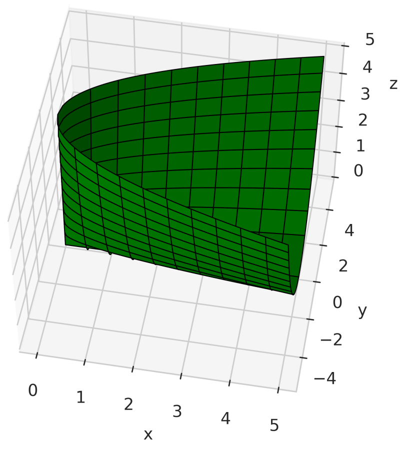


Figure 3.1: Visualization of the cone of 2x2 symmetric positive definite matrices. For this visualization we characterize the SPD matrices by the positivity of the principal minors. Thus for a flattened matrix with entries $(x, y, y, z)^T$, we plot the surface defined by $x > 0$ and $y^2 = xz$.

This inner product induces the Frobenius norm

$$||A||_F = \sqrt{\operatorname{Tr}(A^T A)} = \sqrt{\sum_{i,j=1}^{d} A_{ij}^2}. \tag{3.7}$$

The geodesics in the Euclidean case are simply given by the straight lines between two points

$$\gamma_X(tV) = X + tV \tag{3.8}$$

for a starting point and initial velocity $X, V \in \operatorname{Sym}(\mathbb{R}^d)$ and time $t \in \mathbb{R}$.

As the SPD matrices form a convex set, the Euclidean geodesic between two SPD matrices will always be contained in Sym_{++}. However the SPD matrices lack a vector space structure. Neither is the addition of two SPD matrices necessarily SPD nor is the multiplication with a non-positive scalar. However, they do form an open cone, since multiplication with a positive scalar conserves the positive definiteness. Note that the boundary of said cone are exactly the *positive semi-definite* (PSD) matrices

$$\operatorname{Sym}_+ = \{A \in \operatorname{Sym}(\mathbb{R}^d) \mid x^T A x \geq 0, x \in \mathbb{R}^d, x \neq 0\} \tag{3.9}$$

with at least one null eigenvalue.

Although convex operations like taking the mean are closed inside this cone, the set is incomplete under the Frobenius norm. This poses a problem for numerical algorithms as the convergence inside the SPD matrices can not be guaranteed, and matrices with null eigenvalues are always at a finite distance.

During computations, matrices with negative eigenvalues can quickly appear. This has proven a great difficulty for example in the regularization of Diffusion Tensor Images, where the dispersion of water molecules inside human tissue is modelled by a tensor grid of SPD matrices [Pennec et al., 2004]. By physical laws the uncertainty of the position of a water molecule must always be strictly positive and thus the covariance matrix describing their distribution always positive definite. An aspect closely connected to this considers interpolating between two SPD matrices along the geodesics. In their Euclidean representation, this corresponds to linearly interpolating their coefficients. In this case, the determinant of the interpolant matrices become strictly larger than at the endpoints being again physically unrealistic.

A more intricate perspective comes by looking at invariance properties of the Euclidean distance. The Frobenius inner product is invariant to the congruence action of the orthogonal group

$$A.O = OAO^T \tag{3.10}$$

for $A \in \mathrm{Sym}_{++}(\mathbb{R}^d)$ and $O \in \mathrm{O}(\mathbb{R}^d)$. That is,

$$\langle OAO^T, OBO^T\rangle_F = \langle A, B\rangle_F. \tag{3.11}$$

By that the induced geodesic distance, the usual Euclidean distance, is also invariant

$$d_E(OAO^T, OBO^T) = ||OAO^T - OBO^T||_F = ||A - B||_F = d_E(A, B). \tag{3.12}$$

Meaning that measured in the Euclidean distance, the degree of similarity between SPD matrices will not change under orthogonal transformations and patterns detected by our learning algorithms will be recognized in the same manner if we were to transform the whole set.

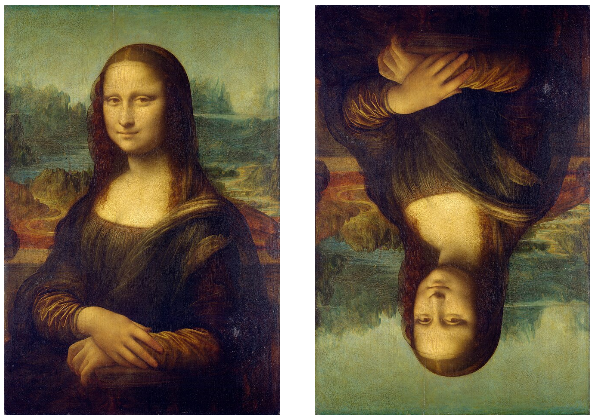


Figure 3.2: Example of an orthogonal transformation. Taking the covariance matrices of image features after such a transformation would result in the same patterns being detected under the Euclidean distance. (Untransformed image from [Leonardo da Vinci, 1503].)

Our goal is to find a geometric characterization of the SPD matrices that captures the information we would like to incorporate into our learning models. To motivate such a structure, we remind ourselves that in the majority of application cases, SPD matrices appear as covariances of some form. Furthermore, it can be shown that covariance matrices are precisely equivalent to positive semi-definite matrices.

Lemma 3.1.1. A matrix is a covariance matrix if and only if it is positive semi-definite.

Proof. By definition $A_{ij} = \text{cov}(x_i, x_j) = \mathbb{E}[(x_i - \mathbb{E}[x_i])((x_j - \mathbb{E}[x_j])]$. So A is symmetric. Further for $w \in \mathbb{R}^d \backslash \{0\}$

$$\begin{aligned} &w^T \mathbb{E}[(X - \mathbb{E}[X])(X - \mathbb{E}[X])]w \\ &= \mathbb{E}[w^T (X - \mathbb{E}[X])(X - \mathbb{E}[X])w] \\ &= \mathbb{E}[(w^T (X - \mathbb{E}[X]))^2] \geq 0 \end{aligned}$$

If A is positive semi-definite, the square root $A^{\frac{1}{2}}$ exists and is nonnegative and symmetric. If X is a random vector with identity covariance matrix then A is the covariance matrix of $A^{\frac{1}{2}}X$. $\square$

Symmetric positive definite matrices can be seen as regularized covariance matrices without null eigenvalues. Note that positive definiteness of a covariance matrix further ensures that no feature can be represented as a linear combination of other features. Because this is assumed to be the case for well-modeled data, covariance matrices are often taken to be non-singular, and thus SPD, leading to a conflation of these notions.

In particular, consider two Gaussian distributions $\mathcal{N}(0, \Sigma_1)$, $\mathcal{N}(0, \Sigma_2)$ completely described by their covariances matrices, which we now assume to be SPD. The swelling effect appears again in the form that the mean of a set of covariance matrices has a larger determinant than the matrices in the set. As the determinants however correspond to the uncertainties of measurements represented by the different Gaussian distributions we find this undesirable. In the same vain, note that the Gaussian distributions are equally specified by their precision matrices Σ_1^{-1}, Σ_2^{-1}. However, the Euclidean distance is generally not invariant under inversion of matrices, just think of strictly positive diagonal matrices for an example. Therefore, we desire a different measure of similarity between SPD matrices, in particular one which balances geometric, invariance and computational aspects.

3.2 Affine-Invariant Riemannian Metric

Now consider a random variable X that is affected by some affine transformation to give a new variable Y

$$Y = AX + b. \tag{3.13}$$

In a real-world application, this might correspond to rigid body motion, for example, changing the constellation of a robot arm [Calinon, 2020]. The mean of the random variables transform in the same way

$$\mu_Y = A\mu_X + b \tag{3.14}$$

Looking at the covariance matrix Σ_X we see that

$$\Sigma_Y = \mathbb{E}[(Y - \mu_Y)(Y - \mu_Y)] = A\Sigma_X A^T. \tag{3.15}$$

This means that, if we want a specific function learned on the covariance matrices to be invariant under affine transformations of the underlying signal, we have to demand our geometry to be invariant to the congruence action of the general linear group

$$\mathrm{GL}(\mathbb{R}^d) = \{M \in \mathbb{R}^{d\times d} \mid \det(M) \neq 0\} \tag{3.16}$$

given by

$$A.M = MAM^T \text{ for all } M \in \mathrm{GL}(\mathbb{R}^d) \text{ and } A \in \mathrm{Sym}_{++}(\mathbb{R}^d). \tag{3.17}$$

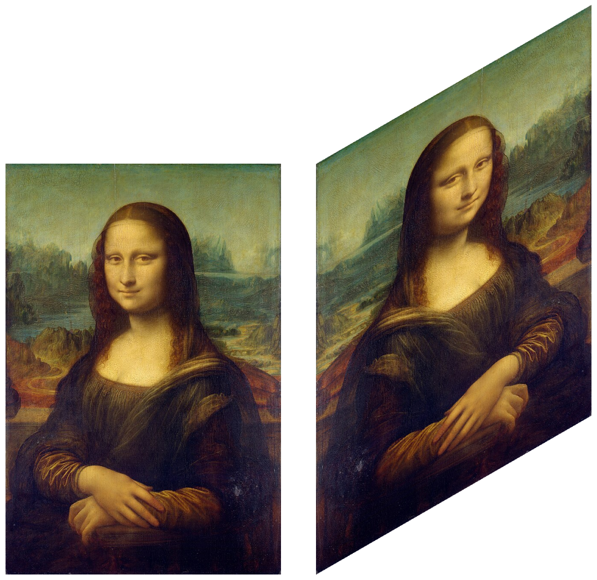


Figure 3.3: Example of an affine transformation. Shearings are an affine transformation that is not part of the orthogonal group. (Untransformed image from [Leonardo da Vinci, 1503].)

At each point $P \in \mathrm{Sym}_{++}(\mathbb{R}^d)$ the tangent space is given by the symmetric matrices $\mathrm{Sym}(\mathbb{R}^d)$. Requiring affine-invariance gives a Riemannian metric defined via the Frobenius scalar product

Definition 3.2.1 (Affine-invariant Riemannian metric (AIRM) [Pennec et al., 2004])**.** Let $A, B \in \mathrm{Sym}(\mathbb{R}^d)$ and $P \in \mathrm{Sym}_{++}(\mathbb{R}^d)$. We define the *affine-invariant Riemannian metric* by

$$\langle A, B\rangle_P = \langle P^{-\frac{1}{2}}AP^{-\frac{1}{2}}, P^{-\frac{1}{2}}BP^{-\frac{1}{2}}\rangle_F. \tag{3.18}$$

Moreover the AIRM is also invariant to matrix inversion.

Starting from a point $P \in \mathrm{Sym}_{++}$ the affine-invariant geodesic going a distance $t \in \mathbb{R}$ into direction $W \in \mathrm{Sym}$ is

$$\mathrm{Exp}_P(tW) = P^{\frac{1}{2}} \exp\left(tP^{-\frac{1}{2}}WP^{-\frac{1}{2}}\right) P^{\frac{1}{2}}. \tag{3.19}$$

Under the AIRM Sym_{++} becomes a Cartan-Hadamard manifold, that is a geodesically complete simply connected Riemannian manifold with nonpositive sectional curvature [Minh and Murino, 2017]. Most importantly we find a unique geodesic connecting any two points. The geodesic distance derived from this is

$$d_{ai}^2(A, B) = ||\log(A^{-\frac{1}{2}}BA^{-\frac{1}{2}})||_F^2. \tag{3.20}$$

We additionally also get an algebraic structure on the SPD matrices. As the action of the general linear group on the SPD matrices is transitive they can be understood as a homogeneous space of the form

$$\mathrm{Sym}_{++}(\mathbb{R}^d) = \mathrm{GL}(\mathbb{R}^d)/\mathrm{O}(\mathbb{R}^d). \tag{3.21}$$

See the direct relation to the polar decomposition of invertible matrices. Every invertible matrix can be written as a composition of a positive scaling along some axis and a subsequent orthogonal transformation of the coordinate system.

As the Hopf-Rinow theorem gives the equivalence of geodesic completeness of a Riemannian manifold with its completeness as a metric space, we get the existence of the AIRM Frechet mean of any finite number of SPD matrices [Fletcher, 2020]. Uniqueness of the mean, however, will only be guaranteed for a geodesic ball of sufficiently small radius.

We formulate our supervised learning framework for SPD-valued functions going into the affine-invariant Riemannian manifold. Imagine again we have function $f\colon \mathcal{X} \to \mathrm{Sym}_{++}(\mathbb{R}^d)$, we assume to data sampled i.i.d. from $\mathcal{X} \times \mathrm{Sym}_{++}(\mathbb{R}^d)$ according to some measure μ. In most practical applications, $\mathcal{X}$ will be Euclidean, and we are trying to approximate a tensor field. Assume additionally our noise to be sampled from a Riemannian normal distribution for which a central limit theorem holds. As we have seen, the natural loss function in this scenario is squared geodesic loss d_{ai}^2. With that

$$\min_{f\in\mathcal{F}} \int_{\mathcal{X}\times\mathrm{Sym}_{++}(\mathbb{R}^d)} d_{ai}^2(f(x), A)d\mu. \tag{3.22}$$

An obstacle is that the expectation of a random variable on the AIRM is only guaranteed to exist if the measure μ has compact support. However, geodesic completeness guarantees the existence and negative curvature the uniqueness of the mean for the compact case. The mean is not available in closed form and can only be approximated iteratively by geodesic descent.

$$\hat{\mu}_{t+1} = \exp_{\hat{\mu}_t}\left(\frac{1}{N}\sum_{i=1}^{N}\log_{\hat{\mu}_t}(A_i)\right) \tag{3.23}$$

for $\{A_1, \dots, A_N\} \subset \mathrm{Sym}_{++}$. Overall the objective function of the empirical risk minimization problem

$$J(f) = \hat{\mathcal{R}}(f, f_0) + \lambda\Omega(f) \tag{3.24}$$

transforms into

$$f_{t+1} = \exp_{f_t}(-\epsilon(\nabla\hat{\mathcal{R}}(f_t, f_0) + \lambda\nabla\Omega(f_t))). \tag{3.25}$$

The affine invariant metric possesses a very regular structure and powerful invariance properties. However, function learning becomes an iterative approximation of tensor fields using gradient flows. Particularly learning on sparsely distributed training data that is not easily confined to a grid structure proves difficult [Pennec et al., 2004].

Another challenge is that due to the coupled matrix inversions in the computation of the affine invariant distance, the calculation quickly becomes very expensive, as the pairwise computation of the distances of N SPD matrices of size $d \times d$ grows with $\mathcal{O}(N^2 d^3)$ [Minh and Murino, 2017].

3.3 Log-Euclidean Metric

We mentioned earlier that the exponential and logarithmic maps are diffeomorphisms between the cone of SPD matrices and its tangent space, the vector space of symmetric matrices. Remarkably these maps are actually global, meaning that it is possible to pull Euclidean vector space operations onto the SPD matrices.

The Log-Euclidean metric was first defined by [Arsigny et al., 2006] by putting on a vector space structure on the SPD matrices. We get a vector space addition

$$\odot : \mathrm{Sym}_{++}(\mathbb{R}^d) \times \mathrm{Sym}_{++}(\mathbb{R}^d) \to \mathrm{Sym}_{++}(\mathbb{R}^d) \tag{3.26}$$

$$A \odot B : \exp(\log(A) + \log(B)) \tag{3.27}$$

and even a scalar multiplication $\mathrm{Sym}_{++}(\mathbb{R}^d)$ by:

$$\circledast : \mathbb{R} \times \mathrm{Sym}_{++}(\mathbb{R}^d) \to \mathrm{Sym}_{++}(\mathbb{R}^d) \tag{3.28}$$

$$\lambda \circledast A : \exp(\lambda \log(A)) = A^\lambda \tag{3.29}$$

Overall, the SPD matrices $\mathrm{Sym}_{++}(\mathbb{R}^d)$ become a vector space isometrically isomorph as a metric space to the Euclidean space $\mathbb{R}^{\frac{d(d+1)}{2}}$. There is even an inner product

$$\langle A, B\rangle_{LE} = \langle\log(A), \log(B)\rangle_F. \tag{3.30}$$

The space is geodesically complete and we can write the Log-Euclidean geodesic joining two points $A, B \in \mathrm{Sym}_{++}$ by

$$\gamma(t) = \exp(\log(A) + t(\log(B) - \log(A))). \tag{3.31}$$

As a vector space, the manifold induced by the Log-Euclidean metric is flat. However, the Log-Euclidean vector space is not fully invariant to affine transformations anymore. Instead, we get invariance to the similarity group: rotations and scalings. In addition, the Log-Euclidean metric is also invariant to inversions.

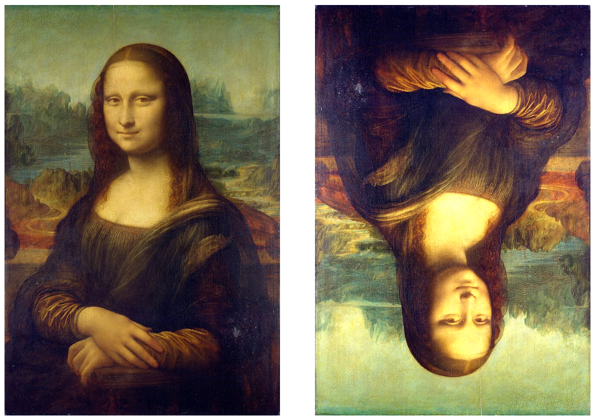


Figure 3.4: Example of a similarity transformation. The scaling in this case corresponds to a change in illumination. (Untransformed image from [Leonardo da Vinci, 1503].)

In the Log-Euclidean domain we get a simple closed-form expression for the empirical mean of a set $X = \{x_1, \dots, x_N\}$

$$\hat{\mathbb{E}}[X] = \exp\left(\frac{1}{N}\log(x_i)\right). \tag{3.32}$$

Euclidean-based statistics can be easily generalized to the Log-Euclidean case with most expressions having a direct equivalent in the logarithmic domain, see [Arsigny et al., 2006] for an exposition.

Using the formula for the affine-invariant geodesic given in 3.19 with the identity as a starting point, we see that the geodesics for the Log-Euclidean and AIRM coincide. The Log-Euclidean metric can thus be seen as a first-order approximation of the affine-invariant metric with respect to curvature. In practice, this view is confirmed, especially for small data scatter, the approximation is good.

The computational complexity of computing the Log-Euclidean distance is much lower than that of the AIRM. This time we can calculate the pairwise distances of a set of N $d \times d$ SPD matrices in $\mathcal{O}(Nd^3)$, see further that the decoupling of the matrices in the distances leads to favorable properties in terms of parallelization [Minh and Murino, 2017].

Note that, due to the logarithm forming a global diffeomorphism between the SPD matrices and the symmetric matrices, we can rewrite a problem of Log-Euclidean function approximation as a problem in the Euclidean space of symmetric matrices

$$\int_{\mathbb{R}^{\frac{d(d+1)}{2}}} ||\exp(y) - \tilde{f}(x)||_F^2 \, d\lambda(x, y) \tag{3.33}$$

where $\tilde{f}(x) = f(\exp(x))$. This is often done in the practical implementation of optimization procedures as the computation of the Riemannian metric involves the directional derivative of the matrix logarithm, and the Riemannian Hessian is generally only available as a finite difference approximation [Han et al., 2021]. As both AIRM and the Log-Euclidean metric can lead to optimization problems that are difficult to solve, there was interest in finding a geometry with favorable optimization properties.

3.4 Bures-Wasserstein

There is yet another geometry we can impose on the SPD matrices. For that, consider again the SPD matrices as a quotient space GL/O. We now view $\mathrm{GL}(\mathbb{R}^d)$ endowed with the restriction of the Frobenius metric. If we consider the map

$$\pi\colon \mathrm{GL}(\mathbb{R}^d) \to \mathrm{Sym}_{++}(\mathbb{R}^d),\ A \mapsto AA^T \tag{3.34}$$

we find that it defines a surjective map onto the SPD matrices. Further, this map defines a Riemannian submersion, which is a map that preserves the metric from one manifold to another. The metric arising from this submersion gives rise to a different Riemannian manifold on the SPD matrices. Note that, however, this time, the manifold is a manifold with a boundary. For the purpose of learning, we will focus on the distance measure of the Bures-Wasserstein geometry.

Take a geodesic in $\mathrm{GL}(\mathbb{R}^d)$ going from a point P in direction V, under the Euclidean metric this just a straight line

$$\mathrm{Exp}_P(tV) = P + tV. \tag{3.35}$$

Using the exposition in [Guigui et al., 2023], we can relate the geodesics in the general linear manifold to those in the SPD matrices

$$\begin{aligned}
\mathrm{Exp}_{AA^T}(tV) &= \pi(A + tS_{AA^T}(V)A) && (3.36)\\
&= (A + tS_{AA^T}(V)A)(A + tS_{AA^T}(V)A) && (3.37)\\
&= AA^T + t(S_{AA^T}(V)AA^T + AA^T S_{AA^T}(V)) && (3.38)\\
&+ t^2 S_{AA^T}(V)AA^t S_{AA^T}(V) && (3.39)\\
&= AA^T + tV + \frac{t^2}{2} S_{AA^T}(V)AA^T S_{AA^T}(V) && (3.40)
\end{aligned}$$

where S_{AA^T} is the solution of the Sylvester equation

$$(AA^T)S + S(AA^T) = V. \tag{3.41}$$

The geodesic distance given by this expression can written in closed form as shown by [Thanwerdas and Pennec, 2023] by

$$d_{BW}(A, B)^2 = \mathrm{Tr}(A) + \mathrm{Tr}(B) - 2\mathrm{Tr}((AB)^{\frac{1}{2}}) \tag{3.42}$$

which is exactly the *Bures-Wasserstein distance* first defined as it applies to SPD matrices in [Bhatia et al., 2019]. Geodesics can, in finite time, hit the boundary. Which in the context of SPD matrices, means that it is possible for matrices to become singular. Looking at the invariance properties of this distance, we find that it is invariant to orthogonal transformations

$$d_{\mathrm{BW}}(A, B) = d_{\mathrm{BW}}(O^T AO, O^T BO) \tag{3.43}$$

for any $O \in \mathrm{O}(\mathbb{R}^d)$. The Bures-Wasserstein distance has an interesting connection to the orthogonal Procrustes problem. Given two SPD matrices A, B, the goal is to find an orthogonal matrix O that minimizes the Frobenius distance between the two matrices. It was shown in [Bhatia et al., 2019], that the minimal distance is equal to the Bures-Wasserstein distance

$$d(A, B)_{BW} = \min_{O \in \mathrm{O}(\mathbb{R}^d)} \|A^{\frac{1}{2}} - B^{\frac{1}{2}} O\|_2. \tag{3.44}$$

It also helps us gain further intuition about the nature of this geometry. It codifies the similarity of full-rank matrices up to rotation.

There is also an interpretation of the Bures-Wasserstein metric as an information geometry expression. Let $\eta = \mathcal{N}(0, \Sigma_\eta)$ and $\nu = \mathcal{N}(0, \Sigma_\nu)$ be multivariate Gaussian distributions with mean 0. If we take $\Gamma(\eta, \nu)$ to be the set of all joint probability distributions with marginals η, ν, we can define the Wasserstein-2 distance

$$\mathcal{W}_2(\eta, \nu) = \left(\inf_{\gamma \in \Gamma(\eta, \nu)} \int_{\mathbb{R}^d \times \mathbb{R}^d} ||x - y||_2^2 \, d\gamma(x, y) \right)^{\frac{1}{2}}. \tag{3.45}$$

The Wasserstein-2 distance of multivariate Gaussians, however, also has a well known description in terms of their mean and covariance

$$\begin{aligned} &\mathcal{W}_2^2(\eta, \nu) && (3.46)\\ &= \|\mu_\eta - \mu_\nu\|_2^2 + \mathrm{Tr}(\Sigma_\eta + \Sigma_\nu - 2(\Sigma_\eta^{1/2} \Sigma_\nu \Sigma_\eta^{1/2})^{1/2}) && (3.47)\\ &= d_{BW}^2(\Sigma_\nu, \Sigma_\eta). && (3.48) \end{aligned}$$

An important aspect is that the manifold defined by the Bures-Wasserstein metric possesses positive curvature, this together with the incompleteness has several major implications. First of all, the Karcher mean of a set of SPD matrices generally only exists in a geodesic ball of small enough radius. Moreover geodesics are no longer unique. Optimization, however, can indeed be easier as a number of cost functions possess favorable convexity properties in this geometry [Han et al., 2021].

The cost of computing the Bures-Wasserstein distances of a set of N SPD matrices are comparable to that of the AIRM scaling with $\mathcal{O}(N^2 d^3)$.

3.5 Conclusion

In this chapter, we looked at symmetric positive matrices and the various geometries we might impose on them. The selection of a suitable geometry for a given application is rather intricate and not self-evident. In some cases, the Euclidean metric will provide excellent and computationally efficient results. Thus, a Riemannian structure on the SPD matrices should not be forced in any case. In other scenarios, we have a phenomenon we want to model precisely. In particular, the AIRM is the natural way to model when working with signals encoded via their Gaussian covariance. The theme of connecting the stochastic world with the geometric one is not confined to the methodology but inherent to the nature of SPD matrices. There is a deep connection between the AIRM and the Fisher information metric, see [Minh and Murino, 2017], which describes the change of information when varying the parameter of a statistical model. In this way, the AIRM provides a unique opportunity to use powerful geometric tools for stochastic reasoning, opening up what is known as information geometry.

The combination of non-compactness and curvature proves to be a significant hurdle, this leads to the development of new geometries which either have a simpler global structure or ease optimization processes. There is yet another host of possible geometries, such as the complete metric space induced by the Stein divergence [Sra, 2016], describing the relations between SPD matrices. Listing and exploring them all would have been beyond the scope of this thesis. However, it is our belief that utilizing ever-newer geometrical structures will also push our understanding of both Gaussian distributions as well as the theory of statistical learning.

We summarize the properties of different metrics in the table below:

Metric	**Structure**	**Complete**	**Isometry**	**Curvature**	**Scale Invariant**
Euclidean	Vector space	No	Orthogonal	Flat	No
Affine-invariant	Manifold	Yes	Affine	Non-positive	Yes
Log-Euclidean	Manifold	Yes	Similarity	Flat	Yes
Bures-Wasserstein	Manifold with boundary	No	Orthogonal	Non-negative	No

Table 3.1: Summary of symmetric positive definite (SPD) metrics and their key properties

Chapter 4

Kernel Methods

The theory of positive definite kernels dates back to the 1950s with the work of [Aronszajn, 1950], while kernel methods in the field of machine learning were originally formulated by [Boser et al., 1992]. Kernel methods were the state-of-the-art in machine learning until recently, when neural networks surpassed them in many areas. Still, kernel theory remains tremendously valuable for learning as it provides a mathematically rigorous background compared to the still lacking analysis of neural networks.

A central feature of kernel methods is the separation of the mathematical object of a positive definite kernel from the algorithm based on matrices of evaluations. This makes the theory of positive definite kernels self-contained; once such a kernel is given, it can be used in any algorithm regardless of the underlying data.

We begin with a brief review of the theory of reproducing kernel Hilbert spaces (RKHS) and scalar kernel methods. We then examine kernels with a SPD domain, before generalizing these to the case of vector-valued RKHS and then attempting to extend the case to SPD-valued functions.

We refer interested readers to [Schölkopf and Smola, 2002] for an in-depth overview of kernel theory and [Álvarez et al., 2012] for a survey on vector-valued RKHS.

4.1 Positive definite kernels

Linear models are well suited for regression and classification tasks, this is due to their flexibility and good optimization properties. However, if the actual function is highly non-linear, our approximation error will be significant. One possible solution is to use a feature map $\phi\colon \mathcal{X} \to \mathcal{H}$ to embed our data into a higher dimensional Hilbert space $\mathcal{H}$, often called the *feature space*, and search for a linear function in this space. Our hypothesis space becomes

$$\mathcal{F}_{\mathcal{H}} = \left\{ f_\theta \colon \mathcal{X} \to \mathbb{R} \;\middle|\; f(x) = \langle \theta, \, \phi(x) \rangle_{\mathcal{H}} \right\}. \tag{4.1}$$

Due to the high, possibly infinite dimension of our feature space $\mathcal{H}$ direct computation is often infeasible. Fortunately, because our functions only access the features through an inner product we can define a *kernel* function allowing us to perform all computations in our original domain $\mathcal{X}$.

Definition 4.1.1 (Positive definite kernel)**.** Let $\mathcal{X}$ be a nonempty set. A *positive definite kernel* is a symmetric function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

$$\sum_{i,j=1}^{N} c_i c_j k(x_i, x_j) \geq 0$$

for all $N \in \mathbb{N}$, $\{x_1, \dots x_N\} \subset \mathcal{X}$ and $\{c_1, \dots, c_N\} \subset \mathbb{R}$.

An equivalent condition is that for any set of points $X \subset \mathcal{X}$ the *Gram matrix*, also called kernel matrix, $K[X]$ defined by $(K[X])_{ij} = k(x_i, x_j)$ is positive semi-definite. For any feature map ϕ, we can define a kernel via

$$k(x', x) = \langle \phi(x'), \phi(x) \rangle_{\mathcal{H}} = \phi(x')^* \phi(x). \tag{4.2}$$

Here ϕ^* denoted the adjoint of the feature map ϕ. A one-to-one correspondence exists between positive definite kernels and *reproducing kernel Hilbert spaces* (RKHS), in which the point evaluation functional is bounded [Aronszajn, 1950]. In that respect, two properties make RKHS especially interesting:

1. For all $x \in \mathcal{X}$, $k(x, \cdot) \in \mathcal{H}_k$
2. If $f \in \mathcal{H}_k$ and $x \in \mathcal{X}$, then $f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}_k}$.

The latter property gives the reproducing kernel Hilbert space its name. The feature map defined by $\phi(x) = k(x, \cdot) = k_x$ is called the canonical feature map.

We then formulate the empirical risk minimization problem for a function defined on the feature space

$$\min_{f \in \mathcal{H}_k} \hat{\mathcal{R}}(f(x_1), \dots, f(x_N)) + \Omega(||f||_{\mathcal{H}_k}) \tag{4.3}$$

where $\Omega\colon \mathbb{R}_+ \to \mathbb{R}$ is a strictly increasing function. The true power of kernel methods now comes from the *representer theorem*, which allows us to reduce the infinite-dimensional optimization problem to a finite-dimensional one, making it numerically tractable.

Theorem 4.1.1 (Representer theorem [Schölkopf et al., 2001])**.** Let $\mathcal{X}$ be a nonempty set, $\{x_1, \dots, x_N\} \subset \mathcal{X}$ and $\mathcal{H}_k$ be a RKHS of functions from $\mathcal{X}$ into the real numbers and k its corresponding kernel. Then the problem 4.3 admits a solution of the form

$$\hat{f}(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i)$$

where $\{\alpha_1, \dots, \alpha_N\} \subset \mathbb{R}$.

Proof. We can decompose any $f \in \mathcal{H}_k$ in a part that lies in $\text{span}(k(\cdot, x_i))$ and the orthogonal complement of that span. That is

$$f = \sum_{i=1}^{N} \alpha_i k(\cdot, x_i) + v$$

such that $\langle k(\cdot, x_i), v\rangle_{\mathcal{H}_k} = 0$ for all i. Using the reproducing property of the kernel we have for any training point x_j

$$\begin{aligned} f(x_j) &= \left\langle \sum_{i=1}^{N} \alpha_i k(\cdot, x_i) + v, k(\cdot, x_j) \right\rangle_{\mathcal{H}_K} \\ &= \left\langle \sum_{i=1}^{N} \alpha_i k(\cdot, x_i), k(\cdot, x_j) \right\rangle_{\mathcal{H}_k} + \langle v, k(\cdot, x_j)\rangle_{\mathcal{H}_k} \\ &= \sum_{i=1}^{N} \alpha_i k(x_j, x_i) \end{aligned}$$

Which means that the loss function in the objective functional is independent of v. For the second part notice that

$$\begin{aligned} \Omega(||f||_{\mathcal{H}_k}) =& \ \Omega\left(\sqrt{||\sum_{i=1}^{N} \alpha_i k(\cdot, x_i)||^2_{\mathcal{H}_k} + ||v||^2_{\mathcal{H}_k}}\right) \\ \geq& \ \Omega\left(||\sum_{i=1}^{N} \alpha_i k(\cdot, x_i)||_{\mathcal{H}_k}\right) \end{aligned}$$

where we use the equation for the first part and the fact that Ω is strictly increasing in the norm for the inequality. With that, it is clear that any minimum of the regularization will be attained in the span of the canonical feature maps. Overall we get the desired result. □

Example: Gaussian kernel

The most widely used kernel in practice is the *Gaussian kernel*, also known as the radial basis function (rbf) kernel.

Definition 4.1.2 (Gaussian kernel)**.**

$$k(x, y) = \exp\left(-||x - y||_2^2 \, / \, 2\sigma^2\right), \tag{4.4}$$

where $\sigma > 0$ is a hyperparameter often referred to as the *bandwidth.* The central feature of this kernel is that it is only dependent on the distance. The associated feature maps embed the data in an infinite dimensional space. An important property of the Gaussian kernel is that it is universal.

Definition 4.1.3 (Universal kernel [Micchelli et al., 2006])**.** Let $\mathcal{X}$ be a domain and $\mathcal{H}_k$ a RKHS of functions on $\mathcal{X}$ with associated kernel k. For a compact subset $\mathcal{Z} \subset \mathcal{X}$ we form the kernel sections spanned by the canonical feature map

$$k(\mathcal{Z}) = \overline{span}\{k_x : x \in \mathcal{Z}\}$$

We call k universal if for any $\mathcal{Z}$, any function $g \in C(\mathcal{Z})$ and any $\epsilon > 0$, we have that there exists a $f \in k(\mathcal{Z})$ such that

$$||g - f||_\infty < \epsilon$$

Note that the above definition is about the approximation in the uniform norm, not the RKHS norm.

4.2 Kernel with SPD domain

One of the strengths of kernel methods is that they are flexible with respect to the domain. In this section, we will look at kernel methods defined on $\mathrm{Sym}_{++}(\mathbb{R}^d)$. The hope is that by defining a positive definite kernel, we can recover many of the powerful machine learning algorithms available in Euclidean space. Essentially any method only relying on a scalar product would be extendable to our manifold case.

When looking for such a kernel, the first question is what kind of functional form we should presume. Since, in many general spaces, only the computation of pairwise distances is possible, the approach of a radial kernel is very natural.

We begin by defining the notion of a conditionally negative (CND) function that will connect the nature of the distance with the properties of an exponential kernel.

Definition 4.2.1 (Conditonally negative definite function)**.** We call a symmetric function $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *conditionally negative* definite if

$$\sum_{i,j=1}^{N} c_i c_j f(x_i, x_j) \leq 0$$

for all $N \in \mathbb{N}$, $\{x_1, \dots x_N\} \subset \mathcal{X}$ and $\{c_1, \dots, c_N\} \subset \mathbb{R}$ **and** $\sum_{i=1}^{N} c_i = 0$.

With this definition we can state a result dating back to [Schoenberg, 1938] giving us a positive definiteness condition for a general kernel of an exponential form.

Theorem 4.2.1 ([Schoenberg, 1938])**.** The exponential kernel

$$k_{\exp(f)}(x, y) = \exp\left(-\sigma f(x, y)\right)$$

is positive definite for all $\sigma > 0$ if and only if f is negative definite.

Our wish is to generalize kernel methods to the SPD matrices. In particular, we want to relate the similarity measured in the feature space to the constraints encoded in the geodesic distance. Thus the Gaussian kernel is the instrument of choice. There is, however, a limitation given by the following theorem:

Theorem 4.2.2 ([Jayasumana et al., 2015])**.** For a metric space (M, d) the kernel $k(x, y) = \exp(-\sigma d^2(x, y))$ is positive definite for all $\sigma > 0$ if and only if there is a Hilbert space $\mathcal{H}$ and a function $\psi\colon M \to \mathcal{H}$ such that $d(x, y) = ||\psi(x) - \psi(y)||_{\mathcal{H}}$.

If M is a complete Riemannian manifold with a geodesic distance d on M, then k is positive definite if and only if M is isometric to a Euclidean space $\mathbb{R}^m$.

Note that the Hilbert space embedding ψ need not be the feature map. A particular implication is that Gaussian kernels based on geodesic distances are only positive definite if the underlying space is flat [Feragen et al., 2015].

Based on that, we see that the Gaussian kernel based on the AIRM (affine-invariant Riemannian metric) is not positive definite for every $\sigma > 0$, because it defines a nonpositively curved manifold. At the time of writing, there is no known result whether it is positive definite for some σ. On the other hand, the Gaussian based on the Log-Euclidean distance is positive definite. By choosing $\psi = \log$ we define an isometry to a Euclidean space.

The Gaussian kernel not being positive-definite for all positive bandwidth parameters σ is a substantial restriction as it prohibits the use of automatic optimization of hyperparameters, which often relies on an assumption of continuous dependence. There is research in finding bands in which the σ reliably gives positive definite kernels on manifolds [Feragen and Hauberg, 2016]. However, the line of research will result in the design of specialized methods for each kind of distance, giving up much of the generality that characterizes kernel approaches.

Fundamentally the positive definiteness conditions of kernel functions defined on a geodesically complete Riemannian manifold reduces to the question of whether an isometric embedding, in the sense of metric spaces, into a Hilbert space is possible. From there, the question arises of how far a kernel approach to learning on flat Riemannian manifolds is advantageous. Depending on the specific algorithm at hand, the isometry of the distance might imply that the optimization problem is identical to a Euclidean one. However, a nonlinear metric will almost certainly be harder to optimize.

An alternative is given by viewing the SPD matrices as a manifold with boundary and utilizing the Bures-Wasserstein distance. As was shown by [Han et al., 2021], the Gaussian kernel with the squared Bures-Wasserstein distance is indeed positive definite. Because this fact is less well-known, we restate the proof for completeness.

Theorem 4.2.3 ([Han et al., 2021])**.** The Gaussian kernel induced by the squared Bures-Wasserstein distance $k(x, y) = \exp\left(-\sigma d_{\mathrm{bw}}^2(x, y)\right)$ is positive definite on the SPD matrices.

Proof. Let $N \in \mathbb{N}$, $\{c_1, \dots, c_N\} \subset \mathbb{R}$ with the condition $\sum_{i=1}^N c_i = 0$ and $\{A_1, \dots, A_N\} \subset \mathrm{Sym}_{++}(\mathbb{R}^d)$.

$$\begin{aligned}\sum_{i,j=1}^N c_i c_j d_{\mathrm{bw}}^2(A_i, A_j) &= \sum_{j=1}^N c_j \sum_{i=1}^N c_i \mathrm{Tr}(A_i) + \sum_{i=1}^N c_i \sum_{j=1}^N c_j \mathrm{Tr}(A_j) \\ &\quad - 2\sum_{i,j=1}^N c_i c_j \mathrm{Tr}\left(A_i^{1/2} A_j A_i^{1/2}\right)^{1/2} \\ &= -2\sum_{i,j=1}^N c_i c_j \mathrm{Tr}\left(A_i^{1/2} A_j A_i^{1/2}\right)^{1/2} \\ &\leq 0.\end{aligned}$$

The statement now follows from 4.2.1. □

The reason we can define a positive definite kernel in this case is because we gave up geodesic completeness - some geodesics might end at the boundary. To the best of our knowledge, there has yet to be any exploration of the Bures-Wasserstein kernel in the context of learning with SPD matrices. Through our numerical experiments in Chapter 6, we will investigate its potential and compare it to the Log-Euclidean kernel and the Euclidean kernel, which are far more widespread in applications.

4.3 Kernel methods

In this section, we will look at different algorithms that utilize positive definite kernels. As mentioned previously, numerical methods generally work directly on the Gram matrix. This gives kernel methods enormous flexibility. The technique being agnostic to the kernel that generates the matrix implies that we can use the same numerical procedure for any set on which we can define a valid kernel.

4.3.1 Support vector classifier

We will now define one of the most popular kernel methods, the support vector classifier (SVC). Assume a binary classification task on $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{\pm 1\}$. Once more, we start from the affine model class. Imagine we want to learn a function that codifies a hyperplane in the data space, which separates our two target classes.

$$f_{(w,b)}(x) = \mathrm{sgn}(\langle w, x \rangle + b) \tag{4.5}$$

such that $f(x_i) = y_i$.

The optimal separating hyperplane is defined as the hyperplane that separates two classes and maximizes the margin between the nearest representatives. The optimization problem can be formulated as

$$\min_{w,b} \quad \frac{1}{2}||w||_2^2 \quad \text{s.t. } y_i(\langle w, x \rangle + b) \geq 1, \quad i = 1, \dots, N, \tag{4.6}$$

where N is the number of training examples.

Our data will not always be perfectly separable. To allow for some misclassification, a soft margin can be introduced by adding slack variables $\xi_i \geq 0$. The optimization problem becomes

$$\min_{w,b,\xi} \quad \frac{1}{2}||w||_2^2 + C\sum_{i=1}^{N}\xi_i \tag{4.7}$$

$$\text{s.t.} \quad y_i(\langle w, x\rangle + b) \geq 1 - \xi_i, \quad i = 1, \dots, N, \tag{4.8}$$

where C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing classification error.

Of course, this approach is not directly viable if our data points lie on a manifold, since we have no notion of linearity. This is where the kernel trick comes in. Using an SPD kernel, we can embed our data into a Hilbert space and define the SVC using the scalar product of said space. To this end, we want to reformulate our algorithm into a form where the input points only appear in the form of a scalar product, which we can then replace with a kernel function. This kind of kernelization is applicable to any algorithm that relies on a scalar product. First, we define the Lagrangian of the above problem

$$L((w,b),\xi,\beta,r) = \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\beta_i y_i(\langle w, x_i\rangle + b) - 1 + \xi_i - \sum_{i=1}^{N} r_i\xi_i. \tag{4.9}$$

Using this we can formulate the dual problem as a maximization problem of the constraints. Note that the solution of the two problems is identical for convex problems.

$$\max_{\beta\in\mathbb{R}^N} \sum_{i=1}^{N}\beta_i - \frac{1}{2}\sum_{i,j=1}^{N}\beta_i\beta_j y_i y_j \langle x_i, x_j\rangle \tag{4.10}$$

$$\text{s.t.} \quad 0 \leq \beta_i \leq C \tag{4.11}$$

$$\sum_{i=1}^{N}\beta_i y_i = 0 \tag{4.12}$$

$$\tag{4.13}$$

We take the optimal offset to be

$$b = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \sum_{i=1}^{N}\beta_i y_i \langle x_i, x_k\rangle\right) \tag{4.14}$$

for $k \in \{i \in \{1,\dots,N\} \mid 0 < \beta_i < C\}$. Using the fact that a positive definite kernel k encodes a scalar product and the representer theorem, we can write the optimal decision function in kernalized form as

$$f(x) = \sum_{i=1}^{N}\beta_i y_i k(x, x_i) + b. \tag{4.15}$$

We have seen how formulating a linear classification algorithm in a dual form allows us to generalize it to a nonlinear method. In the next section, we will look at the same idea in a regression context.

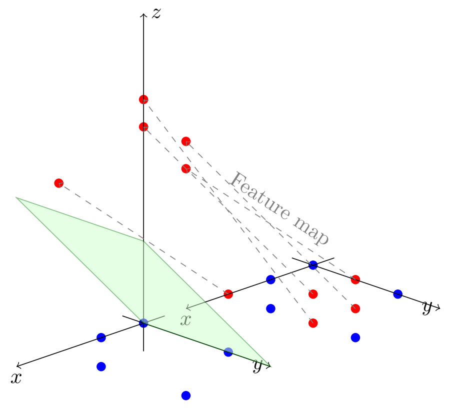


Figure 4.1: Schematic illustration of the kernel trick for SVC. By projecting the data from the plane into a three-dimensional space the two classes, represented by red and blue dots, become linearly separable.

4.3.2 Kernel ridge regression

Kernel ridge regression uses the kernel trick to perform regularized linear regression, also called ridge regression, in the feature space. The estimator function is:

$$\hat{f} = \underset{f\in\mathcal{H}}{\operatorname{argmin}} \left(\sum_{i=1}^{N} (y_i - f(\phi(x_i)))^2 + \lambda \|f\|_{\mathcal{H}}^2 \right), \tag{4.16}$$

where $\lambda > 0$ is a regularization parameter.

From the representer theorem, we know that the solution to the above problem can be found by expressing the hypothesis function in terms of the kernel function as:

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha_i} k(x, x_i). \tag{4.17}$$

From there, we can formulate an objective function for this representation:

$$J(\alpha) = \|y - K\alpha\|_2^2 + \lambda \alpha^T K \alpha, \tag{4.18}$$

where K is the Gram matrix with $K_{ij} = k(x_i, x_j)$. Differentiating with respect to α, we get:

$$\frac{\partial J(\alpha)}{\partial \alpha} = -2K^T(y - K\alpha) + 2\lambda K\alpha = 0. \tag{4.19}$$

Rearranging, we obtain the linear system:

$$(K + \lambda I)\alpha = y. \tag{4.20}$$

We solve the linear system to get the optimal the coefficients

$$\hat{\alpha} = (K + \lambda I)^{-1} y. \tag{4.21}$$

Now that we have defined both classification and regression in a feature space, the next section turns to an unsupervised method where no targets are given.

4.3.3 Kernel PCA

Principal component analysis (PCA) is perhaps the most widely used dimensionality reduction technique. The goal of PCA is to project the data points into a lower dimensional space while preserving as much information as possible. More formally, our goal is to find a set of basis vectors - the *principal components* - that span a subspace that accounts for the maximum amount of variance in the original data. Given a set of centered data points $X = \{x_1, \dots, x_N\} \subset \mathbb{R}^d$ with $\sum_{i=1}^N x_i = 0$ we take the empirical covariance matrix

$$C = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^T \tag{4.22}$$

and solve the eigenvalue problem

$$\lambda v = Cv. \tag{4.23}$$

The eigenvectors given by this problem are then the basis vectors we are looking for. To reduce the data to a certain size n we take the n eigenvectors corresponding to the n largest eigenvalues and project onto the space spanned by them. If we want to kernalize this method we have to reformulate it in a dual way where the input data only appears in a scalar product. Notice first that

$$Cv = \frac{1}{N} \sum_{i=1}^{N} \langle x_i, v \rangle x_i. \tag{4.24}$$

If we now assume that the vector v corresponds to a non-zero eigenvalue it follows directly that it lies in the span of the x_i. We can thus reformulate the PCA as the solution to

$$\lambda \langle x_i, v \rangle = \langle x_i, Cv \rangle. \tag{4.25}$$

PCA is a linear method as it works by projection on a linear subspace. For non-linear relationships, this may not provide good or close to optimal dimensionality reduction. *Kernel PCA* is an extension of the underlying idea of PCA by applying a dimensionality reduction on features in a reproducing kernel Hilbert space. In this case we take the covariance matrix of the feature embeddings

$$C = \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) \phi(x_i)^T \tag{4.26}$$

for an appropriate feature map $\phi\colon \mathcal{X} \to \mathcal{H}$. Inserting into 4.25 we get

$$\lambda\langle\phi(x_i), v\rangle = \langle\phi(x_i), Cv\rangle. \tag{4.27}$$

such that coefficients α_i exist for which

$$v = \sum_{i=1}^{N} \alpha_i \phi(x_i). \tag{4.28}$$

The projection onto the eigenvectors of the feature covariance matrix is finally computed by

$$\langle\phi(x), v\rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x). \tag{4.29}$$

The above shows that RKHS theory opens the possibility to generalize not only supervised learning methods, but also a broader class of algorithms, including dimensionality reduction.

4.4 Vector-valued kernels

Most statistical learning focuses on approximating functions taking scalar values, be it classes encoded with natural numbers or continuous variables in the real numbers. A reason for that is that a vector problem can be seen as the collection of scalar-valued problems, making it possible to focus theory on those atomic tasks. *Vector-valued* function learning has its origins in multitask learning, where the idea is that learning multiple variables at once exploits the correlations between them. It is straightforward to see how such a framework will become relevant in the context of learning covariance matrices.

The theory of vector-valued RKHS, originally developed by [Micchelli and Pontil, 2005], expands the above concept of kernels to matrix-valued kernels. Using those, we can generalize many of the results of the scalar case. First, we will expand the notion of positive definite kernels, to mappings over linear operators. As we are going to restrict ourselves to the finite-dimensional case, the outputs of our new kernels will become matrices. The symmetry condition translates into a requirement on the transpose operator. While the positive definiteness now manifests inside the scalar product of the target space of the RKHS-functions, which is now a vector-valued space.

Definition 4.4.1 (Positive definite matrix-valued kernel)**.** Let $\mathcal{X}$ be a nonempty set. A positive definite *matrix valued kernel* is a function $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{d\times d}$ such that

1.
$$K(x, z) = K(z, x)^T$$

2.
$$\sum_{i,j=1}^{n} \langle y_i, K(x_i, x_j) y_j \rangle_{\mathbb{R}^d} \geq 0$$
for all $n \in \mathbb{N}$, $\{(x_1, y_1), \dots (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y}$.

For each such kernel K there now exists a unique function space $\mathcal{H}_K \subset \mathcal{X}^{\mathbb{R}^d}$, which is a Hilbert space and possesses properties similar to the scalar valued case. In particular we have that

$$K_x \in \mathcal{L}(\mathbb{R}^d, \mathcal{H}_K), \quad x \in \mathcal{X} \tag{4.30}$$

for $K_x = K(\cdot, x)$ and that the space is spanned by all feature maps

$$\mathcal{H}_K = \operatorname{span}\{K_x y \mid x \in X, y \in \mathbb{R}^d\}. \tag{4.31}$$

The representer property for vector valued functions $f \in \mathcal{H}_K$ takes the form

$$\langle f, K(\cdot, x) y \rangle_{\mathcal{H}_K} = \langle f(x), y \rangle_{\mathbb{R}^d}. \tag{4.32}$$

A useful definition is that of separable kernels, for which the calculation of kernel outputs reduces to the scalar case plus a matrix multiplication.

Definition 4.4.2 (Separable kernel)**.** A matrix valued kernel K is called *separable* if there exist a scalar valued kernel k and a positive semi-definite matrix M such that $K(x, x') = k(x, x')M$.

The solution to the vector-valued empirical risk minimization problem in such a reproducing kernel Hilbert space

$$\min_{f \in \mathcal{H}_K} \hat{\mathcal{R}}(f(x_1), \dots, f(x_N)) + \Omega(||f||_{\mathcal{H}_K}) \tag{4.33}$$

is equivalently given by a finite dimensional form stated with kernel evaluations

$$\hat{f}(x) = \sum_{i=1}^{N} K(x_i, x) y \tag{4.34}$$

where y is a vector in $\mathbb{R}^d$.

The presentation above refers to the case where we learn unconstrained functions taking values in a multi-dimensional, but Euclidean space. The construction of methods taking values in constrained sets is more intricate. A central observation is that the kernel is decomposable into the feature maps

$$K(x, y) = K_x^* K_y. \tag{4.35}$$

Now, a question comes up: Can we restrict the operation in the vector-valued RKHS by an additional operator in-between to impose structure on the output of the kernel? The answer to that question is a qualified yes. As we will see in the following, it is possible to derive structured outputs by imposing geometrical constraints in the infinite-dimensional Hilbert space of functions.

4.4.1 Operator theory preliminaries

To define structured matrix kernels, we begin by reviewing background material from operator theory. We will later build upon that to define SPD-valued kernels. In the following $\mathcal{Y}$ will denote a Hilbert space. Linear bounded operators can, in many ways, be seen as the infinite-dimensional generalization of matrices. The adjoint operator is the unique operator commuting in the scalar product. As the adjoint corresponds to transposition, a self-adjoint operator does to a symmetric matrix. Moreover, our familiar notion of positive definiteness generalizes in just the same way.

Definition 4.4.3 (Adjoint, self-adjoint, and positive operators)**.** Let $A \in \mathcal{L}(\mathcal{Y})$. Then:

1. A^*, the *adjoint operator* of A, is the unique operator in $\mathcal{L}(\mathcal{Y})$ that satisfies
$$\langle Ay, z\rangle_{\mathcal{Y}} = \langle y, A^* z\rangle_{\mathcal{Y}}, \text{ for all } y \in \mathcal{Y},\, z \in \mathcal{Y}.$$
2. A is *self-adjoint* if $A = A^*$.
3. A is (strictly) *positive* if it is self-adjoint and for all $y \in \mathcal{Y}$, $\langle Ay, y\rangle_{\mathcal{Y}} \geq 0$ (> 0). We will call the cone of positive operators $\mathrm{Sym}_+(\mathcal{Y})$.

The trace is a measure of the total influence an operator has across all dimensions of a Hilbert space. For matrices, the trace is simply the sum of the diagonal elements. The condition for an operator to be trace-class ensures that the trace is finite.

Definition 4.4.4 (Trace-class)**.** We say $A \in \mathcal{L}(\mathcal{Y})$ is *trace-class* if for any orthonormal basis $\{e_k\}_{k=1}^{\infty}$ of $\mathcal{Y}$

$$||A||_{\mathrm{Tr}} = \sum_{i=1}^{\infty} \langle e_k, Ae_k\rangle_{\mathcal{Y}} = \sum_{i=1}^{\infty} \langle e_k, (A^*A)^{\frac{1}{2}} e_k\rangle_{\mathcal{Y}} < \infty. \tag{4.36}$$

For those we define the trace by

Definition 4.4.5 (Trace)**.** If A is trace-class we define the *trace* by

$$\mathrm{Tr}(A) = \sum_{i=1}^{\infty} \langle e_k, Ae_k\rangle_{\mathcal{Y}}. \tag{4.37}$$

The Hilbert-Schmidt norm is the generalization of the Euclidean matrix norm to infinite dimension. In finite dimensions, linear operators, given by matrices, always have finite Frobenius norm, this is not the case anymore for general operators.

Definition 4.4.6 (Hilbert-Schmidt operators)**.** We say $A \in \mathcal{L}(\mathcal{Y})$ is a *Hilbert-Schmidt* operator if for any orthonormal basis $\{e_k\}_{k=1}^{\infty}$ of $\mathcal{Y}$

$$||A||_{\mathcal{HS}} = \sqrt{\mathrm{Tr}(A^*A)}. \tag{4.38}$$

The space of Hilbert-Schmidt $\mathcal{HS}(\mathcal{Y})$ operators forms a Hilbert space under the scalar product

$$\langle A, B\rangle_{\mathcal{HS}} = \mathrm{Tr}(A^*B) = \sum_{i=1}^{\infty}\langle Ae_i, Be_i\rangle. \tag{4.39}$$

For a compact operator K with countable spectrum $\{\lambda_i\}_{i=1}^{\infty}$ with $\lim_{i\to\infty}\lambda_i = 0$ we have

$$||K||_{\mathcal{HS}}^2 = \sum_{i=1}^{\infty}\lambda_i^2 < \infty. \tag{4.40}$$

If we take the topological product $\mathcal{Y}^d$, we can define linear operators on that space by building matrices of operators, called *block operator matrix.*

Definition 4.4.7 (Block operator matrix)**.** Let $d \in \mathbb{N}$. Then:

1. $A \in \mathcal{L}(\mathcal{Y}^d)$, given by

$$A = \begin{bmatrix} A_{11} & \dots & A_{1d} \\ \vdots & & \vdots \\ A_{d1} & \dots & A_{dd} \end{bmatrix}$$

where each $A_{ij} \in \mathcal{L}(\mathcal{Y})$, $i, j = 1, \dots, d$, is called a *block operator matrix.*

2. The adjoint of A is the block operator matrix $A^* \in \mathcal{L}(\mathcal{Y}^d)$ such that $(A^*)_{ij} = (A_{ji})^*$.

3. A is self-adjoint if $A = A^*$.

4.4.2 Kernel sum-of-squares

In this section, we will present the kernel sum of squares method of [Muzellec et al., 2022] to learn functions constrained to the SPD manifold with boundary. The authors developed a model that utilizes a positive operator in a vector-valued RKHS to constrain the learned function to the bounded SPD cone.

The idea of a kernel sum-of-squares is derived from the sum-of-squares criterion for the positivity of polynomials. In analogy, [Marteau-Ferey et al., 2020] use a similar form to enforce a pointwise non-negativity constraint on the output of kernel methods. Central to their approach is that they aim to construct a universal model that keeps the convexity of the learning problem. [Muzellec et al., 2022] expanded the model then to the case of positive semi-definite matrix outputs.

Learning SPD matrix-valued functions is a vector-valued problem, therefore we can build upon the theory introduced in the previous section. However, instead of searching in the whole space, we will look for a function inside a constrained set. We formulate the risk minimization problem

$$\inf_{\substack{f\in\mathcal{X}^{\mathcal{Y}} \\ f(x)\in\mathrm{Sym}_+}} \quad \mathcal{R}(f) + \Omega(f). \tag{4.41}$$

The first difficulty is to construct a model class guaranteed to take values in the desired set, while keeping many of the properties that make kernel methods appealing. Given a scalar feature map $\phi\colon \mathcal{X} \to \mathcal{H}$ we define our model as

$$F_A(x) = \Phi(x)^* A \Phi(x), \text{ with } A \in \mathrm{Sym}_+(\mathcal{H}^d), \tag{4.42}$$

where $\Phi(x) \in \mathcal{L}(\mathbb{R}^d, \mathcal{H}^d)$ corresponds to

$$\Phi(x) = \begin{Bmatrix} \phi(x) & 0_{\mathcal{H}} & \dots & 0_{\mathcal{H}} \\ 0_{\mathcal{H}} & \phi(x) & \dots & 0_{\mathcal{H}} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{\mathcal{H}} & 0_{\mathcal{H}} & \dots & \phi(x) \end{Bmatrix}. \tag{4.43}$$

Note that A is a block operator matrix. An equivalent representation is given by

$$F_A = \begin{bmatrix} \langle \phi(x), A_{11}\phi(x)\rangle_{\mathcal{H}} & \dots & \langle \phi(x), A_{1d}\phi(x)\rangle_{\mathcal{H}} \\ \vdots & \ddots & \vdots \\ \langle \phi(x), A_{d1}\phi(x)\rangle_{\mathcal{H}} & \dots & \langle \phi(x), A_{dd}\phi(x)\rangle_{\mathcal{H}} \end{bmatrix} \tag{4.44}$$

We have that this model class is linear in its parameter, and constrained to the symmetric positive semi-definite matrices.

Theorem 4.4.1 ([Muzellec et al., 2022])**.** Given $A, B \in \mathcal{L}(\mathcal{H}^d)$ and α, $\beta \in \mathbb{R}$ it holds $F_{\alpha A+\beta B}(x) = \alpha F_A(x) + \beta F_B(x)$. Moreover, if $A \in \mathrm{Sym}_+(\mathcal{H}^d)$ then $F_A(x) \in \mathrm{Sym}_+(\mathbb{R}^d)$ for all $x \in \mathcal{X}$. If the feature map ϕ corresponds to a universal kernel, this model is able to approximate PSD-valued functions arbitrarily well.

The hypothesis class defined by 4.44 gives rise to the optimization problem

$$\inf_{A \in \mathrm{Sym}_+(\mathcal{H}^d)} \hat{\mathcal{R}}(F_A(x_1), \dots, F_A(x_n)) + \Omega(A) \tag{4.45}$$

For the following we will give an explicit regularization map

$$\Omega(A) = \lambda_1 \mathrm{Tr}(A) + \frac{\lambda_2}{2} ||A||^2_{\mathcal{HS}} \tag{4.46}$$

with two regularization parameters $\lambda_1, \lambda_2 \geq 0$. The above map can be seen as a generalization of the elastic net regularization to the operator case.

A very interesting observation first pointed out in [Marteau-Ferey et al., 2020] is that the original point-wise constraints on the outputs become constraints on the parameters of the model. This fundamentally derives from those function constraints defining an infinite dimensional manifold.

Theorem 4.4.2 (Representer theorem for SPD matrix outputs [Muzellec et al., 2022])**.** The empirical risk minimization problem for functions taking values in the SPD manifold with boundary 4.45 has a solution A^*, which may be written as

$$A^* = \sum_{i,j=1}^{N} \Phi(x_i) C_{ij} \Phi(x_j)^*,$$

where $C_{ij} \in \mathbb{R}^{d\times d}$, $i, j \in [N]$, and

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1N} \\ \vdots & \ddots & \vdots \\ C_{N1} & \cdots & C_{NN} \end{bmatrix} \in \operatorname{Sym}_+(\mathbb{R}^{Nd}).$$

The version of the kernel sum-of-squares presented here is separable by $\Phi(x_i)^*\Phi(x_j) = k(x_i, x_j)\mathrm{I}$ for a scalar kernel k. We can evaluate the optimal solution F_{A^*} by means of the k

$$F_C(x) = \sum_{i,j=1}^{N} k(x_i, x)k(x_j, x)C_{ij}, \quad x \in \mathcal{X}. \tag{4.47}$$

The derivation of the finite dimensional representation involves multiple steps and is quite technical, so we only state the result. For convex empirical risk $\hat{\mathcal{R}}$ and $\lambda_2 > 0$ the optimal parameter A^* is unique. After a reparametrization of the pd-constraint matrix C, and the feature map Φ, the problem can be reduced to solving a finite-dimensional matrix problem

$$\min_{B\in\operatorname{Sym}_+(\mathbb{R}^{Nd})} \hat{\mathcal{R}}(F_B(x_1), \ldots, F_B(x_N)) + \Omega(B) \tag{4.48}$$

where for a new finite dimensional feature map Ψ the model is expressed by $F_B(x) = \Psi(x)^T B\Psi(x)$.

The representer theorem allowed us to reduce an infinite dimensional problem on the cone of positive operators into finite dimensional problem on the cone of SPD matrices plus the boundary of singular matrices. This makes the problem numerically tractable, but we still need to respect the manifold condition in our optimization methods, which can be expensive. There is a remedy to this, however, if our risk function fulfills the classical conditions of being convex, lower semi-continuous and bounded from below, we can formulate a dual problem defined over the vector space of symmetric matrices. This then allows us to use unconstrained first-order methods.

First we define the negative part of a symmetric matrix by means of the singular value decomposition. Let $W \in \operatorname{Sym}(\mathbb{R}^d)$, then

$$[W]_- = O^T \max(0, -D)O \tag{4.49}$$

where is the maximum is taken elementwise over the eigenvalues of W. For convex risk $\hat{\mathcal{R}}$ we can define a dual formulation using the notion of a Fenchel conjugate, which is given by

$$\hat{\mathcal{R}}^*(W) = \sup_{A\in\operatorname{Sym}(\mathbb{R}^d)^N} \left\{ \sum_{i=1}^{N} \langle A_i, W_i\rangle_F - \hat{\mathcal{R}}(A) \right\} \tag{4.50}$$

for $W \in \operatorname{Sym}(\mathbb{R}^d)^N$. Using this we can get a dual formulation of the kernel sum-of-squares problem.

Theorem 4.4.3 ([Muzellec et al., 2022]). Let $\hat{\mathcal{R}}\colon \mathrm{Sym}(\mathbb{R}^d)^N \to \mathbb{R}$ be convex, lower semi-continuous and bounded from below and Ω the elastic net regularizer as defined in 4.46. Assume that 4.45 admits a feasible point, and denote $\hat{\mathcal{R}}^*$ the Fenchel conjugate of $\hat{\mathcal{R}}$. Then, If $\lambda_2 > 0$ and $\lambda_1 \geq 0$, 4.45 has the following dual formulation:

$$\max_{W \in \mathrm{Sym}(\mathbb{R}^d)^N} \hat{\mathcal{R}}^* - \frac{1}{2\lambda_2} \left\| \left[\sum_{i=1}^{N} \Psi_i W_i \Psi_i^T + \lambda_1 I \right]_- \right\|_F^2 \tag{4.51}$$

where $W_i \in \mathrm{Sym}(\mathbb{R}^d)$, $i \in [N]$ denotes the i-th matrix element of $W \in \mathrm{Sym}(\mathbb{R}^d)^N$. Further, if W^* is an optimal solution of 4.51, an optimal B^* for the primal problem 4.45 is

$$B^* = \frac{1}{\lambda_2} \Psi^{-1} \left[\sum_{i=1}^{N} \Psi_i W_i^* \Psi_i^T + \lambda_1 I \right]_- \Psi^{-T}. \tag{4.52}$$

Using both the finite-dimensional form of the model and the dual problem we can set up the least squares regression problem on the manifold of SPD matrices including their boundary. For that, we take training points $x_1, \ldots, x_N \in \mathbb{R}^p$ and labels $A_1, \ldots, A_N \in \mathrm{Sym}_+(\mathbb{R}^d)$. The least squares problem on positive semi-definite matrices is given by

$$\min_{B \in \mathrm{Sym}_+(\mathbb{R}^{Nd})} \frac{1}{2N} \sum_{i=1}^{N} \|F_B(x_i) - A_i\|_F^2 + \lambda_1 \mathrm{Tr} B + \frac{\lambda_2}{2} \|B\|_F^2 \tag{4.53}$$

with regularization parameters $\lambda_1 \geq 0$ and $\lambda_2 > 0$.

We now use 4.4.3 to give a dual formulation of this problem

$$\min_{W \in \mathrm{Sym}(\mathbb{R}^d)^N} -\frac{N}{2} \sum_{i=1}^{N} \|W_i\|_F^2 + \langle W_i, A_i \rangle_F + \frac{1}{\lambda_2} \left\| [\sum_{j=1}^{N} \Psi_j W_j \Psi_j^T + \lambda_1 \mathrm{I}]_- \right\|_F^2 \tag{4.54}$$

with the primal-dual optimality relation given by

$$B^* = \frac{1}{\lambda_2} \Psi^{-1} \left[\sum_{i=1}^{N} \Psi_i W_i^* \Psi_i^T + \lambda_1 \mathrm{I} \right]_- \Psi^{-T}. \tag{4.55}$$

The optimization problem in 4.54 is smooth and strongly convex, leading to the existence of a unique minimum. Since it is now also unconstrained in the vector space of symmetric matrices, we are able to use accelerated Nesterov gradient descent. Given symmetric matrices as initialization points, the gradients will be symmetric so that the iterative procedure will not leave the space. This is in stark contrast to the Log-Euclidean least squares regression, which we could define similarly to 3.33 by looking for targets in the form of $\exp(f(x))$. In this case, our regression task is also defined for some unconstrained function f going into the symmetric matrices. However, this kind of parameterization leads to a problem that is non-convex and NP-hard.

Although the kernel sum-of-squares model gives outputs for the less constrained case of Bures-Wasserstein parameterisation of SPD matrices - namely the Euclidean metric submerged into the cone of PSD matrices - we might still expect that this model will also be able to approximate the geometry defined by the affine invariant metric. We therefore test the ability of this regression model to interpolate the affine invariant Riemannian geodesic in Chapter 6.

An interesting question now is whether we can constrain the output of such a model to the interior of the cone. A first naive attempt would be to require our block operator matrix to be strictly positive. However, in infinite dimensions, strict positivity and positive definiteness are not the same anymore. To be able to learn within a closed set, we might shift the class of operators we consider to a regularized version on which the Log-Hilbert-Schmidt distance [Minh et al., 2014] is bounded. For these operators, we gain a vector space structure very similar to the finite-dimensional Log-Euclidean case.

Definition 4.4.8 (Positive definite Hilbert-Schmidt operators)**.**

$$\mathrm{Sym}_{++}(\mathcal{H}) = \{A + \mu I > 0 \colon A = A^*, A \in \mathcal{HS}(\mathcal{H}), \mu \in \mathbb{R}\} \tag{4.56}$$

We define the extended Hilbert-Schmidt norm by

$$||A + \mu I||^2_{\mathcal{HS}_+} = ||A||^2_{\mathcal{HS}} + \mu^2 \tag{4.57}$$

and building on that the Log-Hilbert-Schmidt distance just as in the finite dimensional case

$$d_{log\mathcal{HS}}(A + \mu_1, B + \mu_2) = ||\log(A + \mu_1) - \log(B + \mu_2)||_{\mathcal{HS}_+}. \tag{4.58}$$

We conjecture that this structure might correspond to a linearization similar to the finite-dimensional case, which opens the possibility of a representer theorem. The infinite-dimensional structure of positive definite operators and covariance operators is a very active area of research and entering it will be left for future work. The interested reader is advised to look at [Larotonda, 2007] for an exploration of the concept.

4.5 Conclusion

The theory of reproducing kernel Hilbert spaces is one of the central underpinnings of statistical learning theory. They have proven powerful in practical applications, and the rigorous theory provides a framework for the mathematics of learning. Kernel methods however are still intimately tied to the notion of linearity and depend on our data having a meaningful representation in a Hilbert space. As we have seen in this chapter, even for a regular manifold such as the Cartan-Hadamard SPD matrices, defining a kernel is not possible for every geometry. A certain kind of linearization, either by the first-order approximation of the Log-Euclidean metric or by the imposition of Euclidean topology as done with the Bures-Wasserstein metric enables us to transfer many algorithms to the manifold case by performing them in its feature space.

The case of reproducing kernel Hilbert spaces of functions going into manifolds is even less explored. Here, we considered what, to the best of our knowledge, is the first universal model for structured matrix kernels. Although designed for the least constrained of our geometries, it provides a powerful tool and a promising avenue for further research into approximating functions taking values in non-linear spaces.

In the next section, we will look at a less mature but perhaps even more powerful framework, that of neural networks. Neural networks connect to kernel methods in the way that now we do not fix the kernel, and thus the feature map, a priori, but have the algorithm itself choose the embedding. The hope is that the resulting technique adapts better to variations in smoothness in a problem.

Chapter 5

Deep Learning

Today most state-of-the-art machine learning algorithms in various domains, such as computer vision, natural language processing, or generative models, are based on *neural networks*. The term neural network refers to a family of different parametric machine learning models. The name is derived from its similarity to how networks of neurons propagate information in form of electrical signals in the human brain. Each neuron receives a number of electrical signals and depending the strength of signals and connections either passes the signal to the next neuron or suppresses it.

Similarly, at the basic level, neural networks are characterized by having multiple *layers*, that is being composed of concatenated functions, each with a different set of parameters. In the following we will focus on so-called *feed forward networks*, which are networks in which information is strictly passed from one layer to the next, without data skipping a layer or flowing backwards.

In the first section we will introduce the model class of two layer networks and look at its approximation properties. Afterwards we elaborate on the optimization procedure by which neural network models are typically trained. Following that we will look at neural networks defined on SPD-valued data, in particular the architecture of the *SPDnet*. We use [Goodfellow et al., 2016] as reference for Euclidean deep learning.

5.1 The Feed Forward Architecture

To motivate neural networks, we again begin by looking at the (affine-)linear class. It's not without reason that there's a facetious idiom that all machine learning is, in the end, just linear regression. We set up the typical affine-linear function where we first take the scalar product with a weight parameter w and following that add a bias term b

$$f_\theta = \langle w,\, x \rangle + b, \text{ where } \theta = (w, b). \tag{5.1}$$

To use such a model for classification, we can apply a nonlinear *activation function* to the output. For the choice of the Heaviside function

$$\sigma(x) = \begin{cases} 1 & x > 0, \\ 0 & \text{else} \ , \end{cases} \tag{5.2}$$

we gain the single-layer perceptron class

$$\mathcal{F}_{Perc} = \{ f_\theta(x) = \sigma(\langle w, \, x \rangle + b), \ \text{ where } \sigma \text{ is the Heaviside function} \}. \tag{5.3}$$

This first instance of an artificial neural network was first constructed by [Rosenblatt, 1958] for binary classification. However, this class fails to predict non-linearly separable data. We are going to extend the idea of the perceptron class by adding an additional, so called hidden, layer between the layer representing the input and the output of the network. The networks considered will be fully-connected, meaning that each neuron has a connection to all neurons in the subsequent layer. Neural networks have a natural representation as a directed acyclic graph, as visualized in Figure 5.1. The nodes represent the state of the network, with each column comprising a vector. Meanwhile the edges of that graph have weights attached that denote the strength of a connection between nodes.

Let us break up the composition of a two-layer neural network f

1. **Input Layer**: The input data, denoted as $x^{(0)}$, is passed to the input layer f_0. The transformation at this layer is given by

   $$f_0(x^{(0)}) = \langle w_0, x^{(0)} \rangle + b_0$$

   After this (affine-)linear transformation, a non-linear activation function σ is applied element-wise to produce the output of the layer:

   $$x_i^{(1)} = \sigma(f_0(x^{(0)})_i)$$

2. **Hidden Layer**: The output from the input layer, $x^{(1)}$, serves as the input to the hidden layer. The transformation at this hidden layer is

   $$f_1(x^{(1)}) = \langle w_1, x^{(1)} \rangle + b_1$$

   This produces the output of the hidden layer:

   $$x^{(2)} = f_1(x^{(1)})$$

3. **Output Layer**: The output of the hidden layer, $x^{(2)}$, is the final output of the a two-layer network

   $$y = f(x^{(0)}) = x^{(2)}$$

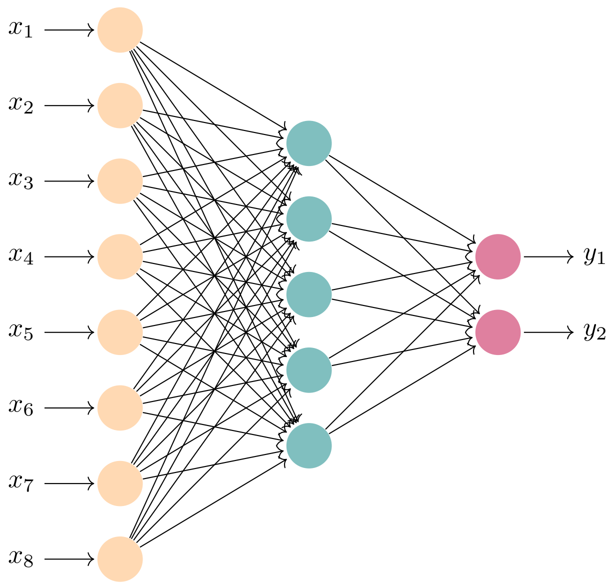


Figure 5.1: Graph representation of a two-layer neural network.

Based on this description we can also write the model class of two-layer neural networks in a more compact form as

$$\mathcal{F}_{\text{2-layerNN}} = \left\{ f_\theta(x) = \sum_{j=1}^{m} a_j \sigma(\langle w,\, x\rangle + b),\, m \in \mathbb{N},\, \theta = (a, w, b) \right\} \tag{5.4}$$

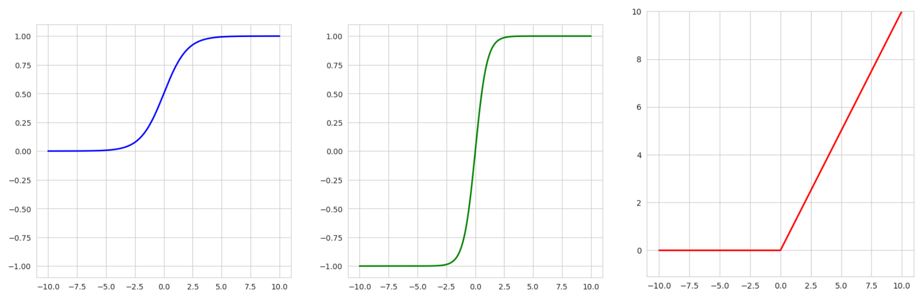
An activation function $\sigma\colon \mathbb{R} \to \mathbb{R}$ is a nonlinear scalar function, which is applied pointwise to the outputs of a, typically linear, neural network layer. The most common are

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad \text{(Logistic Sigmoid)} \tag{5.5}$$

$$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad \text{(Hyperbolic Tangent (tanh))} \tag{5.6}$$

$$\sigma(x) = \max(0, x) \qquad \text{(ReLU - Rectified Linear Unit)} \tag{5.7}$$

Historically, sigmoidal activation functions such as the logistic sigmoid or the hyperbolic tangent have been the most popular choices. However, both suffer from the problem that the norm of the gradient with respect to the parameters can become very large, especially for deeper networks, causing problems during learning. In the modern era of deep learning, ReLU functions have become the most common activation in practice. Since they remain zero for negative inputs, there is a strong tendency for networks to become sparse, with many neurons outputting zeros.



(a) Sigmoid (b) Tanh (c) ReLU

Figure 5.2: Visualization of different activation functions: Sigmoid, Tanh, and ReLU.

A special kind of activation function is given by the *softmax* function. It is typically applied to final output of a neural network in the case of multi-class classification. Assuming we have C classes and using the terminology from above, the outputs become

$$y_i = \text{softmax}(x_i^{(2)}) = \frac{\exp(x_i^{(2)})}{\sum_{j=1}^{C} \exp(x_j^{(2)})}. \tag{5.8}$$

Since each entry lies between zero and one, and the vector sums to 1, the output vector y can now be seen as a probability distribution modeling the chance of an input x belonging to each class.

In general a neural network f_{NN} is built as a concatenation of linear functions combined with pointwise non-linear activation functions. Typically we characterize a network by the number L of its linear layers, often called the *depth*

$$f_{L\text{-layerNN}}(x) = f_L(\sigma(f_{L-1}(\sigma(\cdots f_1(x))))). \tag{5.9}$$

For each layer f_{L-i} we call *width* its number of neurons, which is its domain dimension. In practical applications neural networks can be hundreds to thousands of layers deep.

We now take a look at the approximation theoretic properties of the two-layer neural network class. There is a series of theorems all with slightly different prerequisites and statements that qualify the approximation power of a neural network. We state a basic universal approximation theorem, which shows that the class of two-layer neural networks can approximate any continuous function on a compact set arbitrarily well.

Theorem 5.1.1 ([Hornik, 1991])**.** Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and monotonically increasing function. Then for all $\varepsilon > 0$, all compact sets $U \subset\subset \mathbb{R}$ and all continuous functions $f \in C(U)$, we have that

$$||f - f_{NN}||_\infty < \epsilon$$

for some $f_{NN} \in \mathcal{F}_{2\text{-layerNN}}$.

The theorem establishes that even a simple two-layer neural network architecture holds very powerful approximation properties. Note however, that the statement requires the width of the network to be unbounded. That means that we might have to potentially optimize an extreme amount of parameters, bringing the difficulties of high-dimensional optimization.

5.2 Training neural networks

We are now going to take a closer look at the problem of fitting the parameters of a neural network. Depending on the task at hand, there are essentially two different losses employed. For a regression problem, the objective is given by the familiar Euclidean least squares loss. Our optimization objective for this is

$$\min_\theta J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \|y_i - f_\theta(x_i)\|_2^2. \tag{5.10}$$

On the other hand, when dealing with a multi-class classification problem and using softmax activation, the most commonly used objective function is the cross-entropy loss

$$\min_\theta J(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(f_\theta(x_i))_j \tag{5.11}$$

for a classification problem with C different classes.

Because of the nonlinear activation function, neural network optimization typically becomes a non-convex and challenging optimization problem to solve. We cannot expect convergence to a global optimum, but practice has shown that first-order methods often lead to good local minima.

One of the most popular optimization procedures is the gradient descent algorithm. Gradient descent is based on the idea that being at a point in an optimization landscape we take the current negative gradient, that is direction of the steepest descent, and take a small step in that direction. Given enough such steps, the hope is that we will eventually arrive at a minimum. Assuming we are given an objective function $J(\theta)$, then the gradient descent update rule can be formally written as

$$\theta_{k+1} = \theta_k - \eta\nabla J(\theta_k). \tag{5.12}$$

Where:

- θ_k is the current point.
- η is the *learning rate*. It determines the step size in the direction of the negative gradient.
- $\nabla J(\theta_k)$ is the gradient of J evaluated at θ_k, which points in the direction of the steepest ascent.

The algorithm stops when $\|\nabla J(\theta_k)\|$ is sufficiently small, or after a predetermined number of iterations. In the case of neural networks the size of θ can easily go into the thousands or even millions, making regular gradient descent very computationally expensive. Thus in most cases a variant of *stochastic gradient descent* is chosen instead. In *mini-batch* stochastic gradient descent we choose a random subset $B \subset \{1, \dots, N\}$ of the training data and instead of computing the gradient with respect to the entire training data set we only evaluate the gradient only for this subset

$$\nabla J_B(\theta_k) = \frac{1}{|B|} \sum_{i \in B} \nabla J_i(\theta_k) \tag{5.13}$$

where $\nabla J_i(\theta_k)$ is the gradient for the i-th training point on the k-th iteration of the parameter approximation and update the parameters based on that. Note that for our objective functions J_i corresponds to the single sample loss for a fixed pair (x_i, y_i).

Although originally developed for the purpose of easing the computational burden there is growing evidence that the stochastic nature of the algorithm plays a central role in the success of neural networks. Neural networks are in fact mostly not explicitly regularized, but implicitly regularized by the algorithm itself. The question that remains is how to compute the gradient with respect to

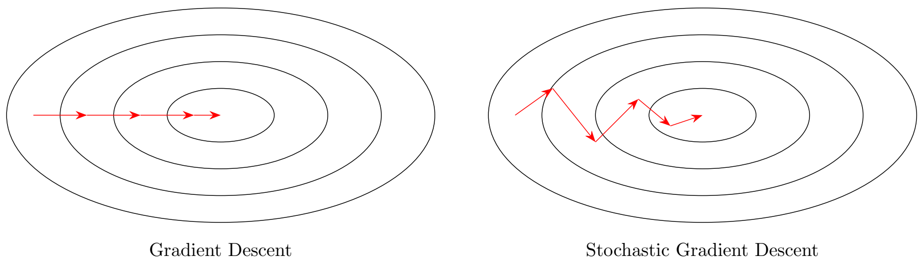


Figure 5.3: Comparison between Gradient Descent and Stochastic Gradient Descent. While gradient descent is more stable and follows a smooth and consistent path to the minimum, stochastic gradient descent is inherently noisy. Gradient descent usually finds a more accurate minimum, while the stochastic version can suffer from oscillations, but is both faster to compute and less memory intensive.

the parameters. We need to adjust the weights of each layer in the direction of steepest descent in parameter space. For large number of parameters and layers, it is paramount that this accomplished in a fashion that avoids unnecessary computations. Note that we can easily compute the gradient of a neural network of depth L with respect to the output layer. For the final layer L, the gradient of the objective J_i with respect to the parameter $\theta^{(L)}$ is:

$$\frac{\partial J_i}{\partial \theta^{(L-1)}} = \frac{\partial J_i}{\partial x^{(L)}} \frac{\partial x^{(L)}}{\partial f^{(L)}} \frac{\partial f^{(L)}}{\partial \theta^{(L-1)}} \tag{5.14}$$

where $x^{(L)}$ is the state after the activation function has been applied to the output of the last layer, and $\frac{\partial x^{(L)}}{\partial f^{(L)}}$ is the derivative of the activation function.

We can then compute the gradients of each layer by recursively stepping backwards through the network, hence the name of the algorithm *backpropagation.* For any layer $L-k-1$, the gradient of the loss with respect to the parameters $\theta^{(L-k-1)}$ of that layer is:

$$\frac{\partial J_i}{\partial \theta^{(L-k-1)}} = \frac{\partial J_i}{\partial x^{(L-k)}} \frac{\partial x^{(L-k)}}{\partial f^{(L-k)}} \frac{\partial f^{(L-k)}}{\partial \theta^{(L-k-1)}} \tag{5.15}$$

while the derivative of the loss with respect to the state $x^{(L-k-1)}$ of the previous layer is

$$\frac{\partial J_i}{\partial x^{(L-k-1)}} = \frac{\partial J_i}{\partial x^{(L-k)}} \frac{\partial x^{(L-k)}}{\partial f^{(L-k)}} \frac{\partial f^{(L-k)}}{\partial x^{(L-k-1)}} \tag{5.16}$$

5.3 Neural networks for data on manifolds

Neural networks have been enormously successful in many domains. However, they rely on the assumption that the input and output data lie in a Euclidean domain. Given that an increasing amount of data is well modeled on other spaces, such as a Riemannian manifold, there has been much interest in extending neural network approaches to such domains. The SPDnet [Huang and Gool, 2017] is one of the first neural network architectures for non-Euclidean data and the most prominent in the context of symmetric positive definite matrices.

We start by assuming that our inputs are defined on a manifold, in our case $\mathcal{X} = \mathrm{Sym}_{++}(\mathbb{R}^d)$ and the output is a real valued scalar y. Our goal is to build a multi-layer network architecture that manages to incorporate the geometric information inherent in the manifold. Since we want this information to be available to all layers, the first challenge is to define functions that can be concatenated while still respecting the manifold structure.

Given the homogeneous space structure of $\mathrm{Sym}_{++}(\mathbb{R}^d) = \mathrm{GL}(\mathbb{R}^d)/\mathrm{O}(\mathbb{R}^d)$, the natural idea is to define the linear part of neural networks as the congruence action of the general group. This leads to the definition of the *Bimap layer*

$$f_W^{(l)}(A) = W.A = W^T A W \tag{5.17}$$

for data $A \in \mathrm{Sym}_{++}(\mathbb{R}^d)$ and learnable weight matrices $W \in \mathrm{GL}(\mathbb{R}^d)$. The manifold assumption in deep learning posits that for typical learning task the data lies on a much lower dimensional manifold. The thought is than that a neural network through consecutive contractive operations is able to capture that structure and exploit it. This heuristic idea is explicit in the construction of the SPDnet, as we known the manifold the data resides upon. As such a Bimap layer can be seen as a map from one SPD manifold to a - typically - lower dimensional more discriminative one. Unfortunately, optimization over the general linear matrices proves difficult as that manifold is non-compact. In [Huang and Gool, 2017], the authors resolved this by restricting the weights to the semi-orthogonal matrices which form a compact manifold $\mathrm{O}(\mathbb{R}^{d_k}, \mathbb{R}^{d_{k-1}})$, thereby defining the central piece of the SPDnet architecture.

The next question coming up is that of non-linearity. Two purposes are served. First, we aim to increase the approximation power of the architecture. Second, the non-linearity is necessary to prevent subsequent layers from collapsing into a single global operation

$$f^{(l)}_{W_3}(f^{(l)}_{W_2}(f^{(l)}_{W_2}(A))) = W_3.(W_2.(W_1.A)) = W'.A \tag{5.18}$$

for some $W' \in \mathrm{O}(\mathbb{R}^{d_3}, \mathbb{R}^{d_0})$.

In analogy to the ReLU function, the authors of the SPDnet decided to define a threshold function on the spectrum of SPD matrices to introduce non-linearity into the network. This happens via means of the *ReEig layer*

$$\mathrm{ReEig}(A) = \sigma(A) = \sigma(O_A D_A O_A^T) = O_A \max(\varepsilon I, D_A) O_A^T \tag{5.19}$$

where the threshold function with threshold value $\epsilon \geq 0$ is defined pointwise on the singular values of the input matrix. A drawback of this approach is that the sparsity-inducing effect of the original ReLU function is mostly lost if the threshold ε is chosen too large.

We might desire, in some cases, to perform additional computations in Euclidean space. For that, we can use the matrix logarithm in form of the *LogEig layer*

$$\mathrm{LogEig}(A) = \mathrm{vec}(O_A \log(D_A) O_A^T) \tag{5.20}$$

to map our SPD data into the space of symmetric matrices and flatten it via the vectorization operation vec. In practice the SPDnet architecture is always used as a neural network with Euclidean output, being trained with either the mean squared error or the cross entropy loss. Thus, there is always a LogEig layer at the conclusion of the Riemannian part of the architecture. This implies that the SPDnet fundamentally operates in a *Log-Euclidean* interpretation of the SPD data.

There are two main challenges in optimizing the SPDnet architecture. One lies in the fact that our parameters lie on a manifold themselves, the manifold of semi-orthogonal matrices, sometimes also called Stiefel manifold. As such using Euclidean methods, in particular the regular Euclidean gradient descent, will almost certain lead to violations of the parameter constrain. Thus we will have to utilize Riemannian optimization. The other difficulty lies the fact that the ReEig and LogEig layer define structured matrix functions necessitating a reformulation of the backpropagation algorithm.

For optimization on manifolds we need a *Riemannian gradient.* We first define the differential on a Riemannian manifold.

Definition 5.3.1 (Riemannian differential)**.** Let $\mathcal{M}$, $\mathcal{M}'$ be two Riemannian manifolds, and let $F\colon \mathcal{M} \to \mathcal{M}'$. We define the differential at a point $p \in \mathcal{M}$ to be the linear map $DF(p)\colon T_p\mathcal{M} \to T_{\mathrm{F}(p)}\mathcal{M}'$ defined by

$$DF(p)[v] = \frac{d}{dt} F(\gamma(t))\bigg|_{t=0} = (F \circ \gamma)'(0), \tag{5.21}$$

where γ is a smooth curve on $\mathcal{M}$, with $\gamma(0) = p$ and $\dot{\gamma}(0) = v$.

Using this differential we can than define the Riemannian gradient.

Definition 5.3.2 (Riemannian gradient)**.** Let $\mathcal{M}$ be a Riemannian manifold. For a smooth function $f \colon \mathcal{M} \to \mathbb{R}$ the Riemannian gradient is the unique vector field $\nabla f(x)$, such that for all points on the manifold $p \in \mathcal{M}$ and all their elements $v \in T_p\mathcal{M}$

$$Df(x)[v] = \langle v, \nabla f(x) \rangle_x. \tag{5.22}$$

With this generalization of a gradient the Riemannian gradient descent is defined, providing a universal scheme for first-order approximation on Riemannian manifolds.

$$\theta_{k+1} = \operatorname{Exp}_{\theta_k}(-\eta \nabla J(\theta_k)) \tag{5.23}$$

The difficulty lies in knowing the gradient efficiently and being able to compute it feasibly. Fortunately in the case of the semi-orthogonal manifold the gradient is known in closed form

$$\nabla_O J^k(W_i^k) = \nabla J^k(W_i^k) - \nabla J^k(W_i^k)((W_i^k)^t W_i^k) \tag{5.24}$$

With this formula we can update the Bimap layer, however, we still need to define a form of derivative for the other two layers. To see this, first notice that both can be understood as functions acting purely on the eigenvalues of the matrix. For such global matrix valued functions the partial derivative defined previously is no longer valid. The theory of *matrix backpropagation* was originally developed by [Ionescu et al., 2015]. In their framework we first compute the linear functional $\mathcal{V}$ giving the variation of the state of one layer x_i with respect to the previous layer state x_{i-1}

$$d\,x_i = \mathcal{V}(d\,x_{i-1}). \tag{5.25}$$

As the tailor expansion of a matrix valued function $f \colon \mathbb{R}^{d \times d} \to \mathbb{R}$ is given by

$$f(x + d\,x) = f(x) + \left\langle \frac{\partial f}{\partial x}, d\,x \right\rangle + \mathcal{O}\left(||d\,x||^2\right) \tag{5.26}$$

we can find the desired partial derivative with the help of the non-linear adjoint given by

$$\mathcal{V}^* \left(\frac{\partial J}{\partial x_i} \right) = \frac{\partial J}{\partial x_{i-1}}. \tag{5.27}$$

This forms a universal scheme that can be applied to all structured matrix functions. For the following expressions we will define two shorthand notations operating on matrices. For a matrix A we will call the systematization

$$\operatorname{sym}(A) = \frac{1}{2}(A^T + A) \tag{5.28}$$

and the diagonalization

$$\operatorname{diag}(A)_{ij} = \begin{cases} A_{ij},\ i = j \\ 0, \text{ else} \end{cases} . \tag{5.29}$$

There is a general form for functions defined on eigenvalue decompositions.

Theorem 5.3.1 ([Ionescu et al., 2015])**.** Let $A \in \text{Sym}_{++}$ have an eigenvalue decomposition $A = ODO^T$. Then the variations with respect to the two components of the decomposition are respectively given by

$$d\,D = \operatorname{diag}(O^T d\,AO) \tag{5.30}$$

and

$$d\,O = 2O(K^T \odot \operatorname{sym}(O^T d\,AO)) \tag{5.31}$$

where $\odot$ is the Hadamard product and K the Loewner matrix operating on the eigenvalues of D

$$K_{ij} = \begin{cases} \frac{1}{\lambda_i - \lambda_j},\ i \neq j \\ 0, \text{ else} \end{cases} . \tag{5.32}$$

The partial derivative of an eigenvalue function is then given by

$$\frac{\partial J}{\partial A} = O\left(2\left(K^T \odot \operatorname{sym}\left(O^T \frac{\partial J}{\partial O}\right)\right) + \operatorname{diag}\left(\frac{\partial J}{\partial D}\right)\right) O^T. \tag{5.33}$$

With the help of this matrix calculus we can finally state the partial derivatives for ReEig layer

$$\frac{\partial J}{\partial O} = 2 \operatorname{sym}\left(\frac{\partial J}{\partial x_k}\right) O \max(\epsilon I, D) \tag{5.34}$$

$$\frac{\partial J}{\partial D} = QO^T \operatorname{sym}\left(\frac{\partial J}{\partial x_k}\right) O \tag{5.35}$$

where Q is a diagonal matrix representing the gradient of $\max(\epsilon I, D)$ given by

$$Q_{ii} = \begin{cases} 1,\ D_{ii} > \epsilon \\ 0, \text{ else} \end{cases} \tag{5.36}$$

and the LogEig layer

$$\frac{\partial J}{\partial O} = 2 \operatorname{sym}\left(\frac{\partial J}{\partial x_k}\right) O \log(D) \tag{5.37}$$

$$\frac{\partial J}{\partial D} = D^{-1} O^T \operatorname{sym}\left(\frac{\partial J}{\partial x_k}\right) O \tag{5.38}$$

as derived in [Huang and Gool, 2017].

5.4 Conclusion

In this chapter we have looked at the model class of neural networks, which through the concatenation of global linear operations with entrywise non-linear activations give a universal approximator for continuous functions. The same principle translated to defining an architecture taking in SPD-valued data and using bilinear layers adapted to respect both symmetry and positive definiteness of the data. This way we were able to propagate data through the network while staying on the manifold. As in the Euclidean case the training procedure was comparatively difficult in contrast to the simplicity of the model class. Both the gradient descent procedure and the computation of the gradient needed to be generalized to the manifold case.

An open question is the construction of multi-layered model classes, such as neural networks, for estimators of manifold-valued functions. As was shown in [Chakraborty et al., 2022] there is a straightforward generalization of the convolution operation often used in neural network architectures to the manifold case by use of the Karcher mean. However, it was also noted that the repeated use of those Karcher mean collapses into a single mean operation in the case of manifolds with constant curvature very similarly as in 5.18. The question of whether this necessarily happens for the case with non-constant curvature remains an open conjecture. For the SPDnet a reduction to a single function was stopped by use of a non-linearity. This might also be a possibility for the case of neural networks taking values in manifolds although it is far from trivial to design effectively nonlinearities.

Chapter 6

Numerical Experiments

This chapter presents several numerical experiments to investigate learning with SPD-valued data. A particular focus will be on exploring the Bures-Wasserstein geometry. This will be done both for encoding functions defined on SPD matrices and using the Bures-Wasserstein ansatz of viewing SPD matrices as a manifold with a boundary to learn functions that go into that cone.

We start by classifying natural images using their covariance descriptors, which define, properly regularized, if necessary, SPD matrices. We then compare the accuracy of exponential kernels defined by the squared Euclidean, Log-Euclidean, and Bures-Wasserstein distances and the SPDnet architecture. Using kernel PCA defined by the respective geometric kernels will provide insight into how the kernels differ in class separation.

Next, we will demonstrate the difficulty of global learning on SPD matrices by comparing the error of kernel ridge regression on sets of SPD matrices with increasing bounds on the maximum eigenvalue.

Finally, we will adapt the experiments of [Muzellec et al., 2022] on interpolating a Bures-Wasserstein geodesic with the kernel sum-of-squares and fit an affine-invariant geodesic instead. The goal is to explore the prospects of using Bures-Wasserstein regression models to approximate affine-invariant relationships.

All kernel methods are based on their scikit learn implementation [Pedregosa et al., 2011] fitted with custom distances, while we use the SPDnet architecture defined in [Brooks et al., 2019] for setting up the neural network part. The kernel sum of squares experiment is based on the source code published by [Muzellec et al., 2022]. Geometric computations are done by the Geomstats library [Miolane et al., 2020b], covariance descriptor preprocessing by [Kongfei, 2020].

6.1 Support Vector Classification

A common task in computer vision is image content classification. A central problem is the high dimensionality of image data; in fact, even a small grayscale image of 1024×1024 pixels has more than a million degrees of freedom when viewed as a vector. In practice, it has been shown that the relevant information is already encoded in a small number of variables. Pixelwise features $f(x, y) \in (\mathbb{R}^{m\times m})^n$ can be intensity, RGB color values, or color gradients. In covariance descriptors [Tuzel et al., 2006], the image is represented by the covariance matrix of these features.

Definition 6.1.1 (Covariance descriptor)**.** Let $X = \{x_1, \dots, x_n\}$ be a set of n features sampled from m^2 pixels and μ_X the mean vector of the features. Then the *covariance descriptor* of an image is given by the covariance matrix of these features

$$\Sigma_X = \frac{1}{n}\sum_{i=1}^{n}(x_j - \mu_X)(x_j - \mu_X). \tag{6.1}$$

The advantages of this representation connect to the geometrical structure of the SPD matrices. Suppose we apply an affine transformation such as a rotation or change in illumination to the image. In this case, this will have no effect on the relations of the covariance descriptors, since an affine change of the underlying random variable corresponds to the congruence action of the general linear group of the covariance matrices. This robustness is very useful in classification, since we assume that the classes should not change just because we rotate the image. However, we have to keep in mind that some information will be lost, since covariance descriptors only encode linear relations.

In the following, we will compare several algorithms, both kernel methods and neural networks, specifically designed to take advantage of the geometric structure of SPD matrices. We will also include a classifier based on the Bures-Wasserstein metric and compare it to SPD methods.

6.1.1 Covariance Descriptors

For our experiments, we choose the well-known ETH-80 dataset [Leibe and Schiele, 2003], which contains 3280 images in 8 categories: apples, cars, cows, cups, dogs, horses, pears, and tomatoes. There are 10 different objects for each category, each taken from 41 different angles, always against a homogeneous blue background. We show four examples from the data set in Figure 6.1. For each image in the data set, we build the covariance descriptors from the feature vector

$$f(x, y) = [x, y, I, |I_x|, |I_y|] \tag{6.2}$$

following [Jayasumana et al., 2015,Minh and Murino, 2017]. Let us break down the features in some more detail. The x and y are giving us the 2D coordinates of the pixels when viewing the image as a plane. I is the intensity at each pixel, while $|I_x|$, $|I_y|$ are the absolute value of the derivative of the intensity

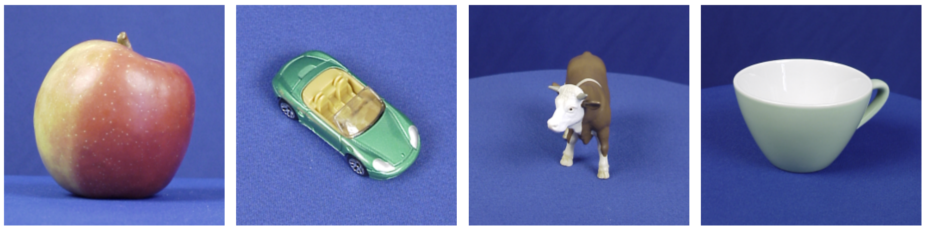


Figure 6.1: Images from the ETH-80 dataset.

in direction of x respectively y. See Figure 6.2 for a showcasing of the image features. Now, we will construct the covariance descriptor of those five feature vectors for each photograph, giving us a data set of 3280 5×5 SPD matrices. To avoid numerical complications from some matrices being close to singularity, we regularize them by adding a small constant value to each eigenvalue of the matrix. Note that during the construction of our data set, the target classes stayed unchanged. We take a preliminary look at the data set using

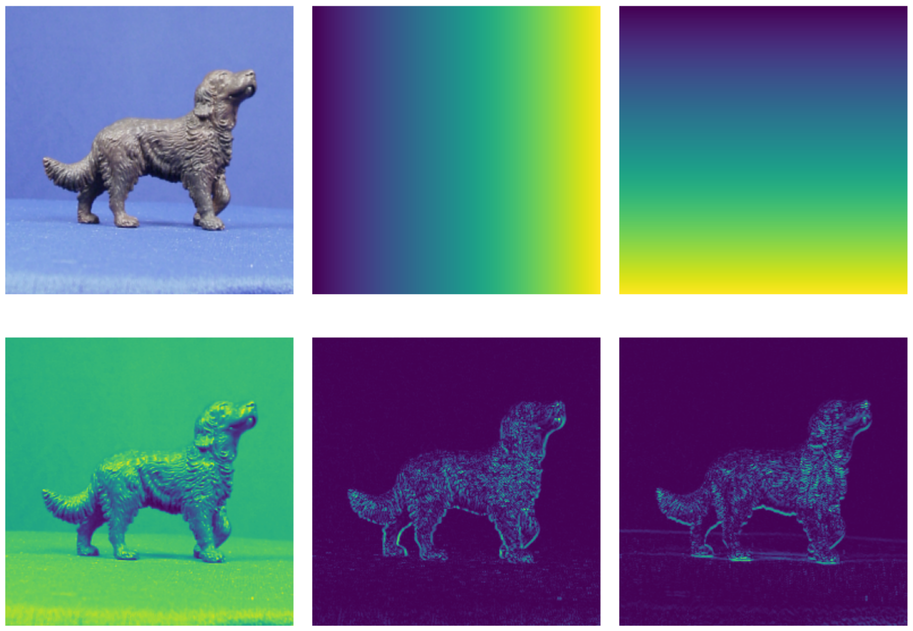
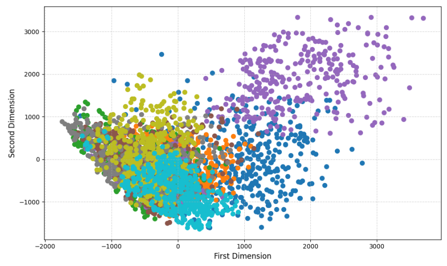
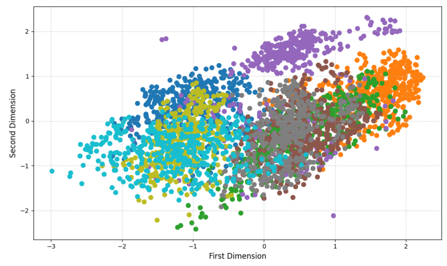


Figure 6.2: An image and its features. From top-left to bottom-right: Original image, x-coordinate, y-coordinate, image intensity ($|I|$), absolute value of intensity derivative in x-direction ($|I_x|$), and absolute value of intensity derivative in y-direction ($|I_y|$).

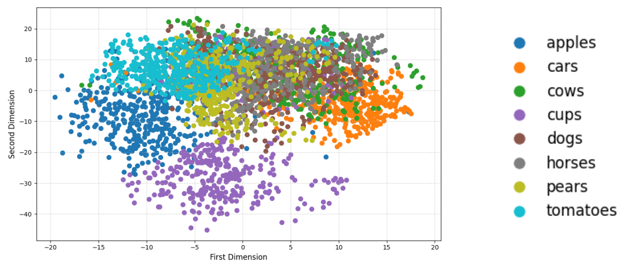
multidimensional scaling (MDS) in Figure 6.3, which takes advantage of the different distances between SPD matrices. MDS is a dimensionality reduction technique that reduces the dimension of the data while trying to preserve the squared distances between data points as much as possible. Unlike PCA, which is essentially a Euclidean technique, MDS allows us to compare the different distances on the original data set. As we can see, the classes as a whole are



(a) Euclidean Distance



(b) LE Metric Distance



(c) BW Metric Distance

Figure 6.3: MDS plots for different distance metrics. The classes are represented by different colors as described in the legend.

clustered relatively tightly, with most classes showing little separability in any of the reductions. The Euclidean distance and the Bures-Wasserstein distance show similar behaviour in that most of the classes cluster very tightly, with the exception of apples and cups, which show more separation and each form an independent cluster. The Log-Euclidean metric, on the other hand, appears to be qualitatively different. There appears to be a line of separation that divides the classes into two categories along what appears to be the diagonal of the two MDS dimensions. The class of cups separates again, but in a much tighter cluster, while the apples now overlap more with the pears and tomatoes. Overall inter-class variability seems comparable in all metrics.

6.1.2 Classification Experiments

We will classify the covariance descriptors using a support vector classifier fitted using a exponential kernel

$$\exp(-\sigma d^2(x,y)) \tag{6.3}$$

with the squared Euclidean, Log-Euclidean, and Bures-Wasserstein distance, respectively. The parameter σ is chosen independently for each kernel using cross-validation. The reason for this is that the scaling parameters are not directly comparable. Matrices that are close at one distance may be far apart at another distance, leading to different clustering and densities. Preliminary tests have shown that using a single cross-validation procedure can cause some kernel methods not to learn at all.

In addition to the kernel methods, we will train an instance of the SPDnet using a cross-validation loss and the ADAM algorithm with a learning rate of 1×10^{-3}. After each epoch we evaluate the network on the test set and save the best parameters.

The accuracy results are reported in Table 6.1. As can be seen, the Euclidean kernel and Log-Euclidean kernels perform minisculy worse than the Bures-Wasserstein kernel. This is a positive result as it shows that the Bures-Wasserstein geometry is able to capture meaningful relationships between SPD matrices. In contrast the SPDnet performs significantly worse than the kernel methods. A possible explanation could be that the SPDnet was originally designed with a strong focus on dimensionality reduction. However, in our use case, the matrices are already quite small, so the SPDnet has little chance of finding a meaningful lower dimensional space. In the following, we will focus

Method	Accuracy (%)
Euclidean Kernel	83
Log-Euclidean Kernel	83
Bures-Wasserstein Kernel	84
SPDnet	65

Table 6.1: Results of the covariance descriptor classification.

on kernel methods and understand how they encode similarity between SPD matrices. Recall that the Gram matrix K of a kernel is a symmetric positive definite matrix generated by evaluating the kernel on each pair of data points. The first thing we do to get a deeper insight into the inner workings of each kernel method is to visualize the computed Gram matrix for each of the three kernels using a heat map in Figure 6.4. As we can see all of the kernels have very full gram matrices signifying they are able to capture a lot of the information in the training set. Zooming in we can see that the diagonal is slightly pronounced which is to be expected as self-similarity of a feature should generally be strong, we do not however, observe any diagonal dominance in the data which might signal overfitting and misaligned hyperparameters. Comparing the three matrices we see more sparsity in the Euclidean and Bures-Wasserstein kernels as opposed to the Log-Euclidean kernel. Further sparse regions appear in similar positions hinting at a possible connection of the feature embeddings.

We are also going to visualize the spectrum of each of these kernel matrices in Figure 6.5. As can be seen from the plot, there are significant differences in the decay of the spectrum. The Bures-Wasserstein kernel has a very slowly decaying

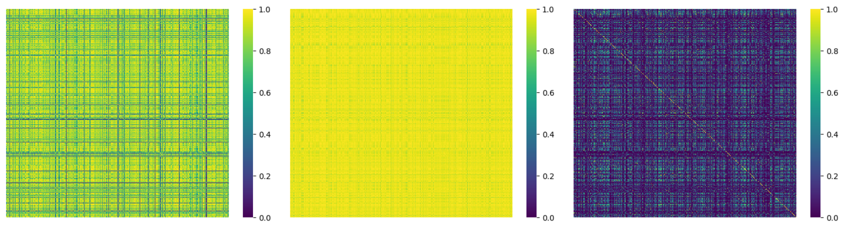


Figure 6.4: Gram matrices visualized: (Left) Euclidean, (Middle) Log-Euclidean, and (Right) Bures-Wasserstein.

spectrum that is well above the others. The Euclidean drops comparatively faster, but much less than the Log-Euclidean kernel. The latter starts to fall very steeply. This could be an indication that the Log-Euclidean kernel is particularly well adapted to the special geometry of SPD matrices.

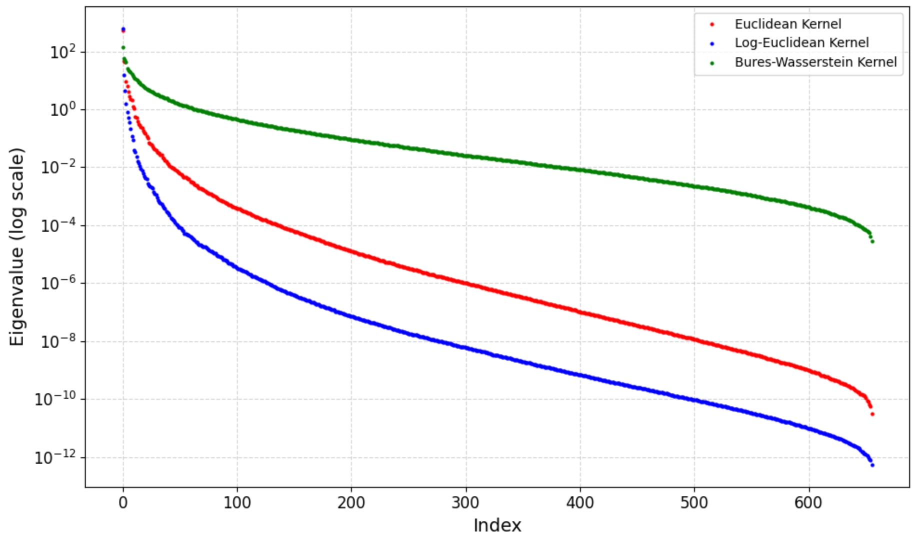
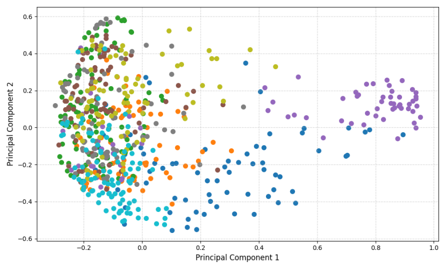
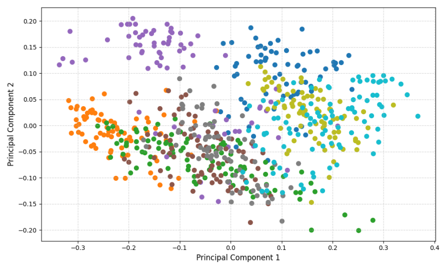


Figure 6.5: Spectrum of the Euclidean, Log-Euclidean and Bures-Wasserstein kernel matrix. The smallest eigenvalue of the Bures-Wasserstein Gramian is given by 2.7×10^{-5}, thus it is positive definite.

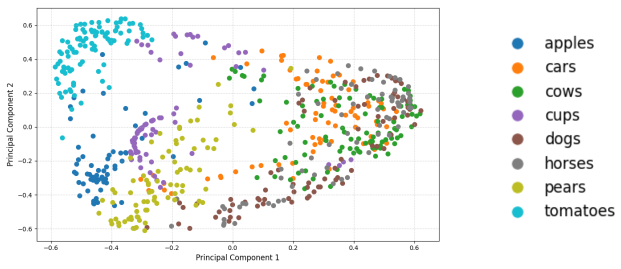
To further compare how the different kernels separate the classes in feature space, we will run kernel PCA on each of them. Here, the covariance is taken in the high-dimensional feature space and then projected down to the two leading principal components so that we can visualize them in a scatterplot in Figure 6.6. We can see that the feature space embedding preserves a significant part of the results of the MDS algorithm. Again, the classes cups, apples, and tomatoes stand out in the Euclidean and Bures-Wasserstein kernels. Only this time they



(a) Euclidean kernel



(b) Log-Euclidean kernel



(c) Bures-Wasserstein kernel

Figure 6.6: Kernel PCA plots showing different distance metrics.

are more separated. In both cases, we can observe that these classes are mainly separated from the other classes by the first principal component. Therefore, they seem to be different in a very significant way. It is not straightforward to interpret the dimensions, but as will be seen more clearly in the following extended experiment, the first principal component here could correspond to a homogeneity of color condition. On the other hand, we see a tight clustering for the other classes, which do not vary much in the first principal component. This behavior is in contrast to the result of the Log-Euclidean PCA, which shows that the variance is much more distributed along the two principal components. Again, we can see the separation line along a diagonal.

Extended feature set experiment

An interesting question that arises is whether adding more features improves classification accuracy. In particular, which kernels are best able to incorporate this additional information. For this next iteration of the experiment, we will construct covariance descriptors based on the feature vector

$$f(x,y) = [x, y, R, B, G, I, |I_x|, |I_y|, |I_{xx}|, |I_{yy}|]. \tag{6.4}$$

As before, we will include the coordinate, intensity, and magnitude of the derivatives. This time, however, we will also include the RBG values at each pixel, as well as the strength of the second derivatives of the intensity. We will visualize these features on another image, in this case a pear, see Figure 6.7. For comparison we will also again plot the MDS reduction for the data set of the covariance descriptors of the extended features in Figure 6.8. This time we see a very clear separation between the Euclidean and Bures-Wasserstein distances and the Log-Euclidean distance. For the Euclidean distance we see a very tight cluster where most of the classes are grouped. However, we see that most of the classes, especially the same classes that stood out in the previous behavior, separate again. Cups, apples, and tomatoes form lines diagonal to the two principal components. A qualitatively similar behavior can be seen in the Bures-Wasserstein case, but here the classes are less tightly packed, suggesting that the Bures-Wasserstein distance might be a little more suitable to explain the differences in the classes. Overall, there seems to be a very pronounced gradient of a latent variable that affects most of the classes, but the fact that the classes are still parallel suggests that they remain quite distinct in high dimensional space. In contrast, the Log-Euclidean MDS plot shows a clear oval shape, with many of the classes seeming to be well separated. However, it is noticeable that the cup class has a very high inter-class variation.

We build the covariance descriptors in the same manner as in the previous experiment. This time we are going to work with 10×10 covariance matrices, again regularized by Tikhonov regularization. The accuracy results are reported in Table 6.2. Several things stand out. All kernel methods perform significantly better, as expected. Furthermore, the geometric kernels separate a bit more from the Euclidean kernel, suggesting that they are able to take better advantage of the additional information provided by the extended features. Overall, the classification accuracy is very high. The SPDnet is still inferior in performance, although the gap has narrowed, suggesting that the primary reason for the lower performance is the low complexity of the data set. Visualizing the Gram

Method	Accuracy (%)
Euclidean Kernel	92
Log-Euclidean Kernel	96
Bures-Wasserstein Kernel	95
SPDnet	86

Table 6.2: Results of the covariance descriptor classification on extended features.

matrices in Figure 6.10 we see the same qualitative behavior as before, only more pronounced this time. The Euclidean kernel show much higher sparsity, with the diagonal clearly visible, while the Log-Euclidean kernel is still very dense.

Looking at the spectral curves in Figure 6.9, we see similar behavior as in the previous experiment. The order in terms of decay remains the same, but this time the Bures-Wasserstein and Euclidean curves are closer, while the gap to the Log-Euclidean kernel is more pronounced.

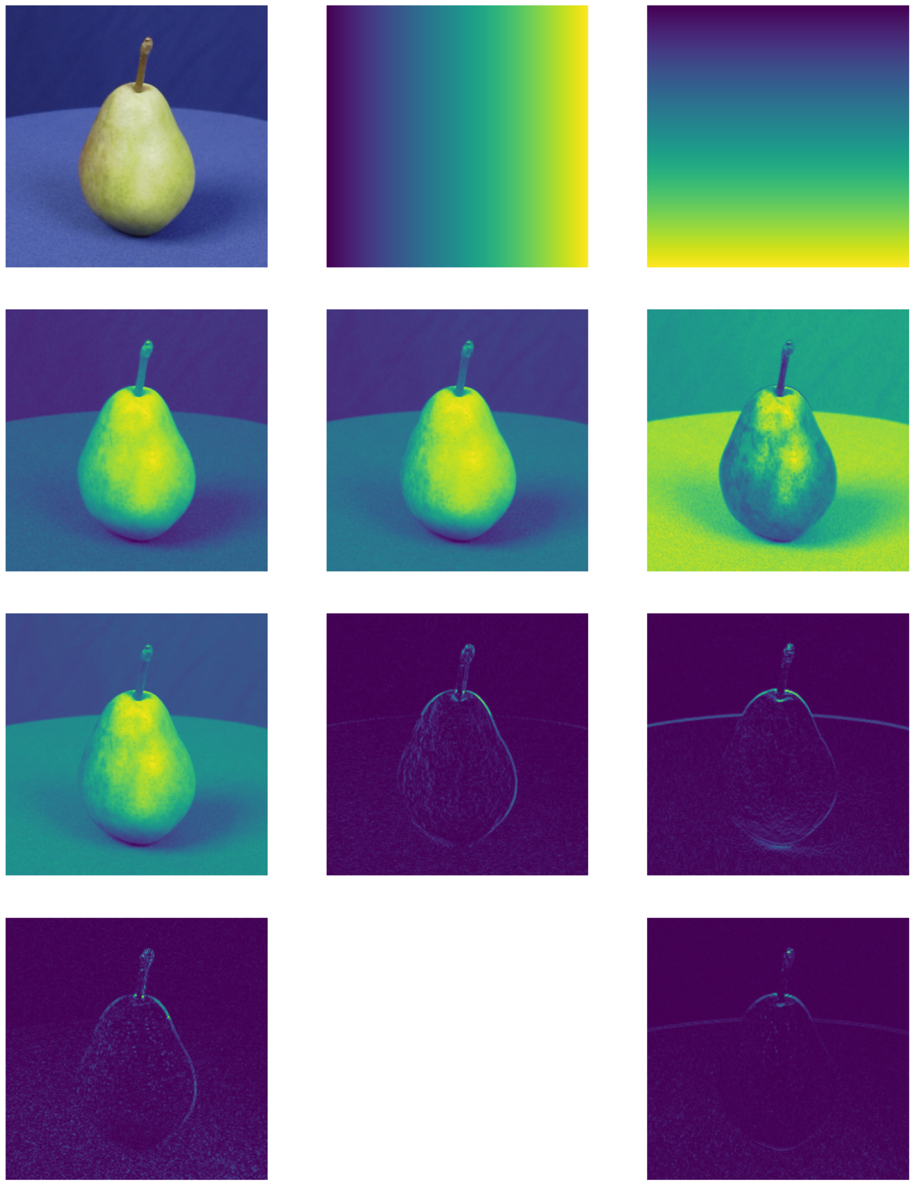
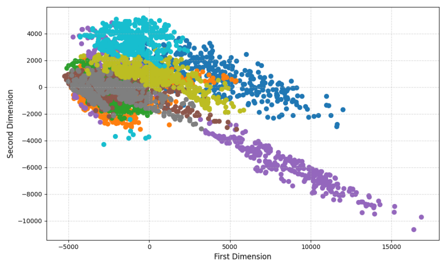
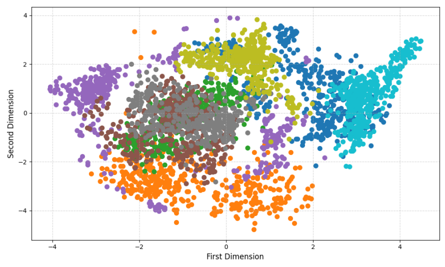


Figure 6.7: An image and its extended features. From top-left to bottom-right: Original image, x-coordinate, y-coordinate, Red channel intensity (R), Green channel intensity (G), Blue channel intensity (B), overall image intensity ($|I|$), absolute value of intensity derivative in x-direction ($|I_x|$), absolute value of intensity derivative in y-direction ($|I_y|$), second derivative in x-direction ($|I_{xx}|$), and second derivative in y-direction ($|I_{yy}|$)

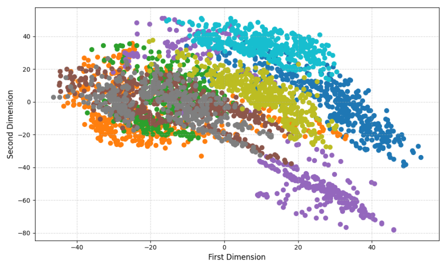
.



(a) Euclidean Distance



(b) LE Metric Distance



(c) BW Metric Distance

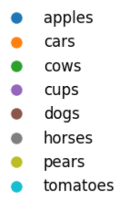


Figure 6.8: MDS plots on the extended descriptor set for different distance metrics.

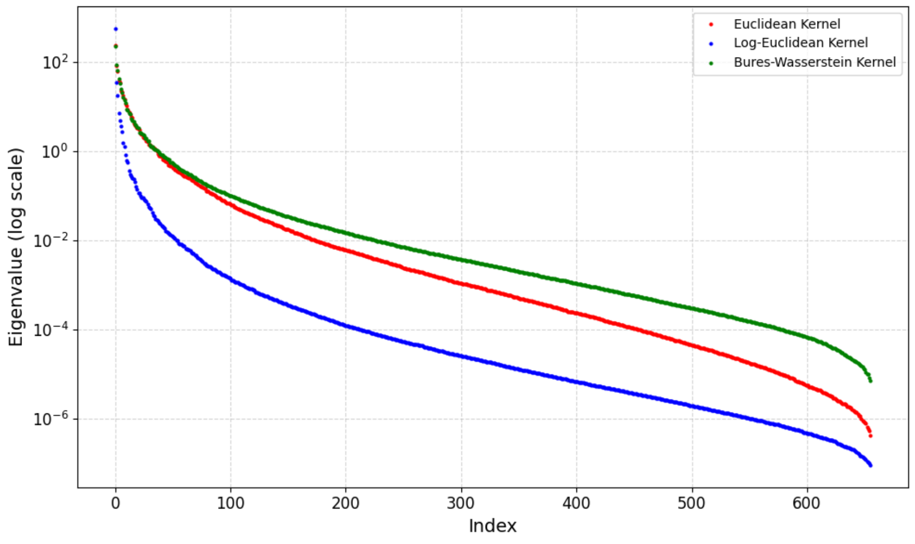


Figure 6.9: Spectrum of the Euclidean, Log-Euclidean and Bures-Wasserstein kernel matrix. The smallest eigenvalue of the Bures-Wasserstein Gramian is given by 7.2×10^{-6}, thus it is positive definite.

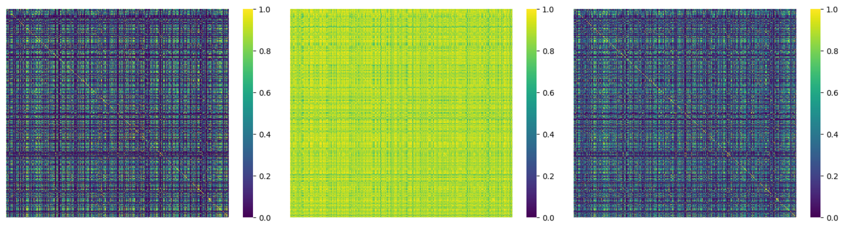
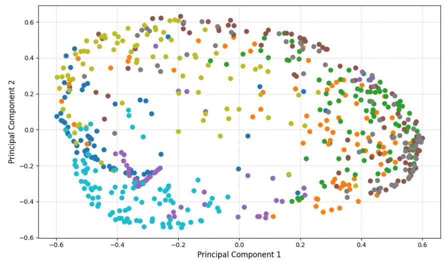
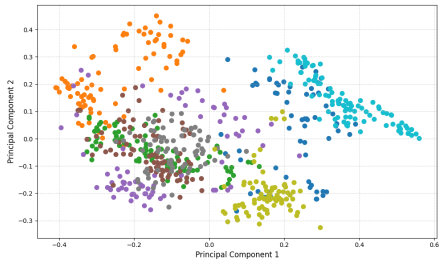


Figure 6.10: Gram matrices visualized: (Left) Euclidean, (Middle) Log-Euclidean, and (Right) Bures-Wasserstein.

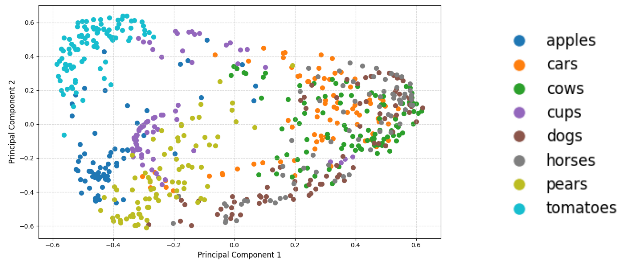
The repetition of the kernel PCA algorithm, as visualized in 6.11 for this experiment gives us further insight. For the Log-Euclidean and Bures-Wasserstein



(a) Euclidean kernel



(b) Log-Euclidean kernel



(c) Bures-Wasserstein kernel

Figure 6.11: Kernel PCA plots showing different distance metrics.

kernels we can see very similar results. The principal component scatter plots are mirror images of each other. We can see that the same classes are separated similarly well. The images containing tomatoes are the most separated followed by apples, in both cases markedly better than in the previous experiments. We surmise that this is because we additionally encoded the *red* values explicitly. The Log-Euclidean kernel PCA shows that the classes are very well separated, with overall much lower inter-class variability than the other two kernels.

6.2 Kernel Ridge Regression

In this section, we will analyze the effect of increasing the size of a bounded set in the SPD matrices on the performance of kernel ridge regression defined with different kernels. The goal is to see how well different kernels perform in a regression setting.

We define the log-det map on the SPD matrices using the affine-invariant Riemannian metric, since this is often what we want to approximate.

$$g(A) = \log(\det(A)) \tag{6.5}$$

which is a function that goes from the SPD matrices to the real numbers. To motivate this task we will follow [Boumal, 2023] and see that it conforms to a form of linearity. For this we need a few definitions, all aiming at using geodesics, interpreted as straight lines, to generalize notions on the manifold. First, convexity is given by

Definition 6.2.1 (Geodesic convexity)**.** A function $f\colon \mathrm{Sym}_{++}(\mathbb{R}^d) \to \mathbb{R}$ is called *geodesically convex*, if $f \circ \gamma\colon [0,1] \to \mathbb{R}$ is convex for every geodesic $\gamma\colon [0,1] \to \mathrm{Sym}_{++}(\mathbb{R}^d)$.

In the general case the definition of geodesic convexity is a little more intricate as geodesics might not exist or be unique between any two points of the manifold. However, due to the very regular structure of the AIRM as a Cartan-Hadamard manifold, the definition simplifies and we do not have to concern ourselves with geodesic convexity for subsets of the manifold. The definition of geodesic concativity is now straightforward

Definition 6.2.2 (Geodesic concativity)**.** A function $f\colon \mathrm{Sym}_{++}(\mathbb{R}^d) \to \mathbb{R}$ is called *geodesically concave*, if $-f$ is geodesically convex.

With both these terms we can then define geodesic linearity

Definition 6.2.3. A function $f\colon \mathrm{Sym}_{++}(\mathbb{R}^d) \to \mathbb{R}$ is called *geodesically linear* if it is both geodesically convex and geodesically concave.

The log-det map g is in fact geodesically linear on the SPD matrices with the affine-invariant metric. A typical assumption in learning with SPD data is that the matrices lie in a compact set of the form

$$\{A \in \mathrm{Sym}_{++}(\mathbb{R}^d) \mid c_1 \leq \lambda_{\min}, \cdots, \lambda_{\max} \leq c_2\} \tag{6.6}$$

that is, sets defined by bounded eigenvalues. In this definition $\lambda_{\min}$ and $\lambda_{\max}$ referred to the minimal and maximal eigenvalue of any matrix in the set. As we discussed earlier, the geometric structure makes global learning difficult. We now generate compact sets of 5×5 SPD matrices with an increasing upper bound on the eigenvalues and their logarithmic determinants as targets. We generate 100 SPD matrices by sampling a diagonal matrix from a multivariate uniform distribution and an orthogonal matrix from the QR decomposition of a $d \times d$ multivariate normal distribution. We fit three exponential kernels with the squared Euclidean, Log-Euclidean, and Bures-Wasserstein distances and perform kernel ridge regression on each. All hyperparameters are set using cross-validation. The result in Figure 6.12 shows that all three kernels have good

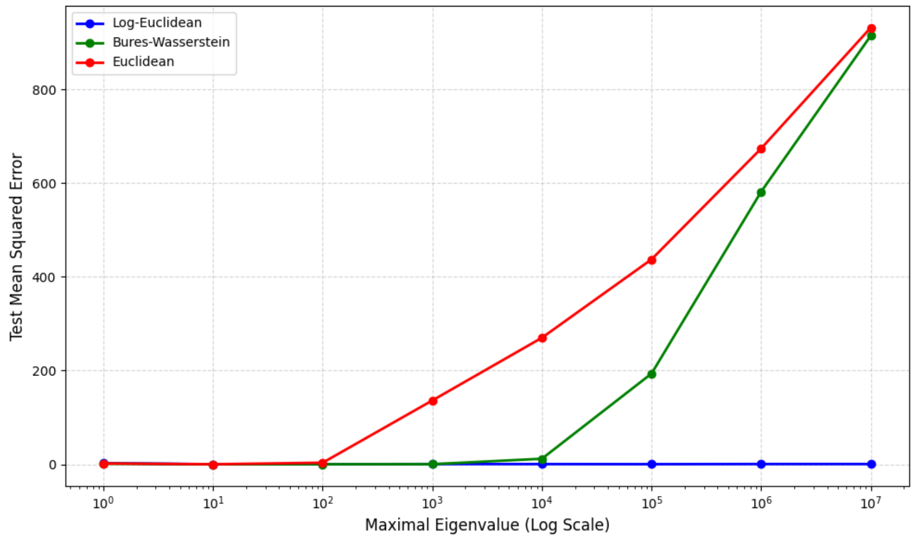


Figure 6.12: Error plot of the kernel ridge regression for the log-det map using different kernels.

approximation properties for sufficiently small bounds. After a while the error of the Euclidean increases exponentially with the bound. The Bures-Wasserstein kernel proves to be a better approximation of affine-invariant linearity, but also suffers from exponentially increasing error for larger bounds, asymptotically converging to the Euclidean model. The Log-Euclidean kernel however does not suffer from increasing error at all and continues to provide a very good approximation of the AIRM even on larger scales.

We can also see how the kernel models adjust their bandwidth parameter σ. For each maximum eigenvalue, we performed another cross-validation grid search to allow each model to find the best hyperparameters. Notice that as the bounds increase, the Euclidean and Bures-Wasserstein kernels choose smaller and smaller σ to compensate, while the Log-Euclidean only adjusted the parameter once, as can be seen in Figure 6.13. The implication of this is again that the log-Euclidean geometry is a true approximation of the affine invariant one, while the others do not fit naturally.

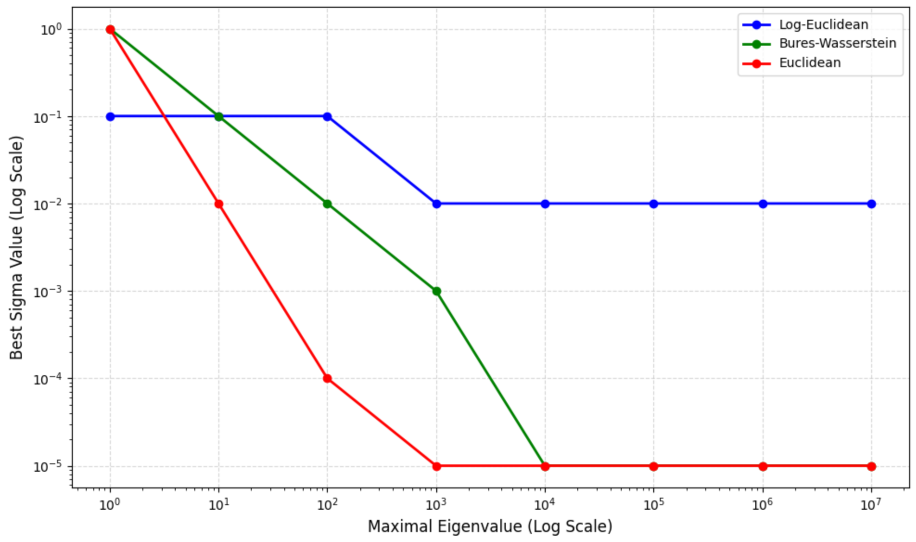


Figure 6.13: Bandwidth plot of the kernel ridge regression for the logdet map using different kernels.

6.3 An outlook to manifold-valued prediction: Kernel sum-of-squares

Next, we present numerical experiments using the PSD-valued kernel sum-of-squares model. In their original paper, [Muzellec et al., 2022] demonstrated the ability of the kernel sum-of-squares model to capture the Bures-Wasserstein geometry by interpolating a geodesic. Motivated by whether the model, originally built to approximate functions going into the SPD manifold with boundary, can also estimate maps going into the affine-invariant geometry, our goal will be to replicate the experiments for an affine-invariant geodesic. To do this, we will interpolate between the two matrices

$$A = \begin{bmatrix} 5 & -2 \\ -2 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 22 & 4 \\ 4 & 1 \end{bmatrix} . \tag{6.7}$$

We have fast convergence to a minimum loss of zero for the kernel sum of squares model, with the gradient norm being zero within 1.00×10^{-7} precision after 23 iterations. However the model is fitted in the Frobenius norm. To get a better intuition of the results the two geodesics are visualized in Figure 6.14. Looking at the predicted geodesic, we see that the determinants of the predicted matrices are not fully linearly interpolated, as is a central goal of the affine-invariant geometry. There is also visible tendency for the matrices to be closer to singularity than in the true geodesic, which could be explained by the Bures-Wasserstein geometry, which includes covariance matrices of less than full rank. This effect is even more pronounced when comparing the true geodesic to

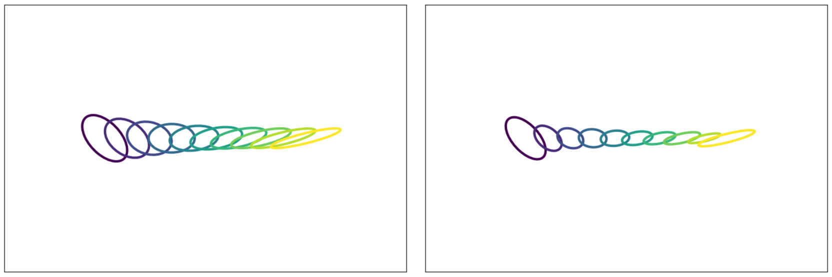


Figure 6.14: The true (left) and predicted affine-invariant geodesic (right). The SPD matrices are visualized as ellipses, where the axis are given by the two eigenvectors of the matrix, while height and width are determined by the eigenvalues. The purple ellipse on the left end represents the matrix A, while the yellow matrix at the right end represents the matrix B. For better visualization we scale the determinants of the matrices by a factor dependent on the largest determinant in the geodesic. Note that the determinant controls the volume of the ellipses.

the predictions on a finer grid with additional points there were not part of the training data, see Figure 6.15. There is a slight oscillation into the direction of singularity of points that were added. To further illuminate on this we repeat

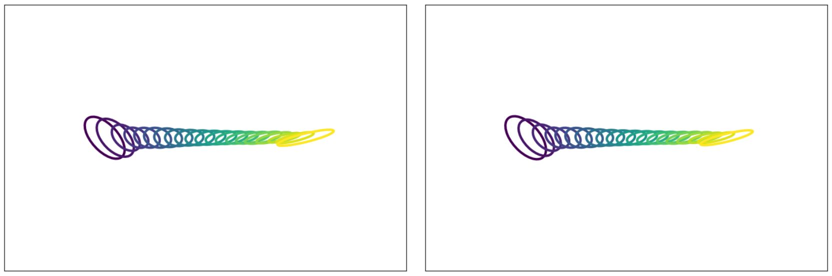


Figure 6.15: The true (left) and predicted affine invariant geodesic (right) on a finer grid.

the experiment, but reduce the bandwidth parameter to consciously induce overfitting. We show the predictions on the same grids in Figure 6.16. While the geodesic on the training points improves and now almost exactly matches the true geodesic, we get a Runge-like phenomenon on the finer grid. The fit in this case is tight at the endpoints, but oscillates strongly in the direction of the cone boundary. Another observation is that the curve also approaches a singularity at the training points near the center of the geodesic. This is quite interesting because it suggests that the function approximation in the Bures-Wasserstein metric may follow laws close to the Euclidean case.

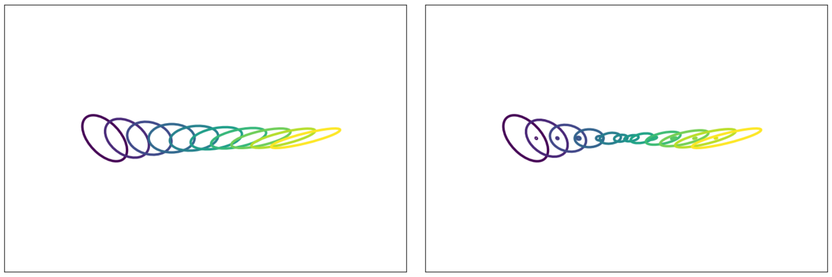


Figure 6.16: The fitted geodesic (left) and the predictions on a finer grid (right) with lower bandwidth. The geodesic now is a very close fit on the training points, but when looking at newly sampled points the matrices become almost singular.

6.4 Conclusion

In this chapter we have presented several numerical experiments aiming to provide further insight into the differences of the various geometries, in particular the geodesic distances. In a classification task on image data we have evaluated the support vector classifiers based on Gaussian kernels with different squared geodesic distances and the SPDnet. We found that all kernel methods performed well, while SPDnet performed mediocrely on the data set with a limited number of features. A key observation is that the Bures-Wasserstein and Euclidean distances lead to very similar behavior, as visualized by both the MDS dimensionality reduction, kernel PCA and the heat map of the trained kernel matrices. The Log-Euclidean metric, on the other hand, differed significantly from both. In general, the Log-Euclidean metric seemed to be much better at capturing the similarity relationships between different SPD matrices, with a much denser heat map and more uniform PCA plots. However, this did not translate into a difference in classification accuracy. One reason may be, as suggested by the spectrum of the Bures-Wasserstein Gramian, that the variation between SPD matrices, especially that between different classes, is already well explained by a small number of variables. Since all kernels embed the data in an infinite dimensional feature space, the differences between the geometries might be nullified as the classes separate very neatly.

The same observation could be made in the kernel ridge regression experiment, where both Bures-Wasserstein and Euclidean kernels showed exponentially deteriorating performance in approximating a function that is geodesically linear in the affine-invariant metric, while the Log-Euclidean metric showed little increase in error. We conjecture that the Bures-Wasserstein metric, defined as a quotient metric derived from a submersion of the Frobenius metric onto the SPD matrices, is fundamentally Euclidean in its function approximation properties. The differences observed in the experiments may be primarily a matter of scaling. The Log-Euclidean metric on the other hand is fundamentally non-

Euclidean and a good approximation of the affine-invariant Riemannian metric.

Nevertheless as we have seen in the sum-of-squares experiment working on the bounded cone of positive semi-definite matrices can be used to approximate SPD-valued functions if only properly tuned. The great advantage of the Bures-Wasserstein metric over the Euclidean case is that while our matrices may become singular, they remain symmetric and PSD. Further research on parameter and error bounds or additional regularization techniques may thus be fruitful.

Chapter 7

Conclusion

This thesis has considered three metrics under which the symmetric positive definite matrices become a Riemannian manifold. The affine-invariant metric defines a geodesically complete manifold with negative curvature and, by its invariance to the isometry group of the homogeneous space structure GL/O, provides the maximal algebraic characterization of SPD matrices. However, due to its non-compact nature, it eludes possible embedding into a reproducing kernel Hilbert space, thus prohibiting the use of kernel methods. Its first-order approximation, the Log-Euclidean metric, and the Bures-Wasserstein metric as the quotient metric of the general linear group with Frobenius metric to the right action of the orthogonal group enable such embeddings.

In our numerical experiments, we have seen that the performance in a classification task slightly favors the geometric approaches, albeit at a higher computational cost. The difference may not be so pronounced because the geometric disparities flatten out in infinite dimensional feature space. Another consideration is that both Log-Euclidean and Bures-Wasserstein geometric are Euclidean on some level. The deeper study of topological and metric space aspects of the learning problem is an area for further research.

In contrast, kernel ridge regression has shown that the geometries are not the same. Taking a problem more tailored to the differential nature of SPD matrices, both the Euclidean and the Bures-Wasserstein kernels showed decreasing performance when approximating on an increasingly larger geodesic ball. Conversely, the Log-Euclidean kernel did not suffer from any degradation. This suggests that the latter is more suitable than the Bures-Wasserstein kernel for learning tasks that require a very close approximation to the affine-invariant Riemannian manifold.

On the other hand, the SPDnet showed reasonable accuracy but underperformed the kernel methods in covariance descriptor classification. It is possible that the data set in this experiment was too small or too simple for the network to reach its full potential. Neural networks tend to perform well on large data sets, while the advantage over kernel methods is generally smaller in the small data regime. Another reason may be that the orthogonal constraints of the parameters proved to be limiting in terms of network expressiveness. Further research could go in the direction of extending the parameters to a larger manifold and finding ways to optimize on the full manifold of the general linear group. Neural networks have shown that even local minima can lead to good performance in practice, so the non-compact nature of the manifold may not be a major obstacle.

Neural networks for functions taking SPD values are still a very open area of research. The lack of vector space structure of sets of manifold valued functions naturally leads to the formulation of maps as weighted averages. The global nonlinearity of manifolds makes learning with multilevel structures difficult, since the concatenation of manifold-valued functions can easily collapse to either a single or a trivial function.

Learning on non-Euclidean spaces is challenging. Our familiar world and intuition begin to fail and must be reconstructed. This brings both hardship and understanding. Rising to the challenge of geometric learning leads to better and more consistent algorithms for new data types, and respecting their structure can inherently lead to more powerful learning schemes. It also serves as a driver for new theory in the search for new ways to look at structure in data.

Bibliography

[Álvarez et al., 2012] Álvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012). Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 4(3):195–266.

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

[Arsigny et al., 2006] Arsigny, V., Fillard, P., Pennec, X., and Ayache, N. (2006). Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421.

[Bhatia et al., 2019] Bhatia, R., Jain, T., and Lim, Y. (2019). On the bures–wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191.

[Bihan et al., 2001] Bihan, D., Mangin, J.-F., Poupon, C., Clark, C., Pappatà, S., Molko, N., and Chabriat, H. (2001). Diffusion tensor imaging: Concepts and applications. *Journal of magnetic resonance imaging : JMRI*, 13:534–46.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA. Association for Computing Machinery.

[Boumal, 2023] Boumal, N. (2023). *An introduction to optimization on smooth manifolds*. Cambridge University Press.

[Brooks et al., 2019] Brooks, D., Schwander, O., Barbaresco, F., Schneider, J.-Y., and Cord, M. (2019). Secondorder networks in pytorch. In *GSI 2019 - 4th International Conference on Geometric Science of Information*, pages 751–758, Toulouse, France.

[Calinon, 2020] Calinon, S. (2020). Gaussians on riemannian manifolds: Applications for robot learning and adaptive control. *IEEE Robotics & Automation Magazine*, 27(2):33–45.

[Chakraborty et al., 2022] Chakraborty, R., Bouza, J., Manton, J. H., and Vemuri, B. C. (2022). Manifoldnet: A deep neural network for manifold-valued data with applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):799–810.

[Devroye et al., 1996] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*, volume 31 of *Stochastic Modelling and Applied Probability*. Springer.

[Feragen and Hauberg, 2016] Feragen, A. and Hauberg, S. (2016). Open problem: Kernel methods on manifolds and metric spaces. what is the probability of a positive definite geodesic exponential kernel? In Feldman, V., Rakhlin, A., and Shamir, O., editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1647–1650, Columbia University, New York, New York, USA. PMLR.

[Feragen et al., 2015] Feragen, A., Lauze, F., and Hauberg, S. (2015). Geodesic exponential kernels: when curvature and linearity conflict. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)*, pages 3032–3042. IEEE. null ; Conference date: 07-06-2015 Through 12-06-2015.

[Fletcher, 2020] Fletcher, T. (2020). 2 - statistics on manifolds. In Pennec, X., Sommer, S., and Fletcher, T., editors, *Riemannian Geometric Statistics in Medical Image Analysis*, pages 39–74. Academic Press.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Adaptive computation and machine learning. MIT Press.

[Grohs, 2012] Grohs, P. (2012). Quasi-interpolation in Riemannian manifolds. *IMA Journal of Numerical Analysis*, 33(3):849–874.

[Guigui et al., 2023] Guigui, N., Miolane, N., and Pennec, X. (2023). Introduction to riemannian geometry and geometric statistics: From basic theory to implementation with geomstats. *Foundations and Trends® in Machine Learning*, 16(3):329–493.

[Han et al., 2021] Han, A., Mishra, B., Jawanpuria, P. K., and Gao, J. (2021). On riemannian optimization over positive definite matrices with the bures-wasserstein geometry. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 8940–8953. Curran Associates, Inc.

[Hornik, 1991] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

[Huang and Gool, 2017] Huang, Z. and Gool, L. V. (2017). A riemannian network for spd matrix learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 2036–2042. AAAI Press.

[Ionescu et al., 2015] Ionescu, C., Vantzos, O., and Sminchisescu, C. (2015). Matrix backpropagation for deep networks with structured layers. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2965–2973.

[Jayasumana et al., 2015] Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. (2015). Kernel methods on riemannian manifolds with gaussian RBF kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(12):2464–2477.

[Jumper et al., 2021] Jumper, J. M., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Zídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D. A., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589.

[Karcher, 1977] Karcher, H. (1977). Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30:509–541.

[Kipf and Welling, 2017] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

[Kongfei, 2020] Kongfei, H. (2020). Covariance-Descriptor. `https://github.com/KongfeiH/Covariance-Descriptor/tree/master`.

[Larotonda, 2007] Larotonda, G. (2007). Nonpositive curvature: A geometrical approach to hilbert–schmidt operators. *Differential Geometry and its Applications*, 25(6):679–700.

[Leibe and Schiele, 2003] Leibe, B. and Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–409.

[Leonardo da Vinci, 1503] Leonardo da Vinci (1503). Italian: Ritratto di monna lisa del giocondo. portrait of mona lisa del giocondo. Painting.

[Marteau-Ferey et al., 2020] Marteau-Ferey, U., Bach, F. R., and Rudi, A. (2020). Non-parametric models for non-negative functions. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[Micchelli and Pontil, 2005] Micchelli, C. A. and Pontil, M. A. (2005). On learning vector-valued functions. *Neural Comput.*, 17(1):177–204.

[Micchelli et al., 2006] Micchelli, C. A., Xu, Y., and Zhang, H. (2006). Universal kernels. *J. Mach. Learn. Res.*, 7:2651–2667.

[Minh and Murino, 2017] Minh, H. Q. and Murino, V. (2017). *Covariances in Computer Vision and Machine Learning*. Springer Cham.

[Minh et al., 2014] Minh, H. Q., San Biagio, M., and Murino, V. (2014). Log-hilbert-schmidt metric between positive definite operators on hilbert spaces. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

[Miolane et al., 2020a] Miolane, N., Guigui, N., Brigant, A. L., Mathe, J., Hou, B., Thanwerdas, Y., Heyder, S., Peltre, O., Koep, N., Zaatiti, H., Hajri, H., Cabanes, Y., Gerald, T., Chauchat, P., Shewmake, C., Brooks, D., Kainz, B., Donnat, C., Holmes, S., and Pennec, X. (2020a). Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9.

[Miolane et al., 2020b] Miolane, N., Guigui, N., Brigant, A. L., Mathe, J., Hou, B., Thanwerdas, Y., Heyder, S., Peltre, O., Koep, N., Zaatiti, H., Hajri, H., Cabanes, Y., Gerald, T., Chauchat, P., Shewmake, C., Brooks, D., Kainz, B., Donnat, C., Holmes, S., and Pennec, X. (2020b). Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9.

[Mohri et al., 2018] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2 edition.

[Muzellec et al., 2022] Muzellec, B., Bach, F., and Rudi, A. (2022). Learning psd-valued functions using kernel sums-of-squares.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

[Pennec, 2006] Pennec, X. (2006). Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25:127–154.

[Pennec et al., 2004] Pennec, X., Fillard, P., and Ayache, N. (2004). A riemannian framework for tensor computing. Technical Report RR-5255, INRIA. inria-00070743.

[Ramesh et al., 2022] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

[Rudi et al., 2018] Rudi, A., Ciliberto, C., Marconi, G., and Rosasco, L. (2018). Manifold structured prediction. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Schoenberg, 1938] Schoenberg, I. J. (1938). Metric spaces and positive definite functions. *Transactions of the American Mathematical Society*, 44(3):522–536.

[Schölkopf et al., 2001] Schölkopf, B., Herbrich, R., and Smola, A. J. (2001). A generalized representer theorem. In Helmbold, D. and Williamson, B., editors, *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[Sra, 2016] Sra, S. (2016). Positive definite matrices and the S-divergence. *Proceedings American Mathematical Society (PAMS)*.

[Steinke et al., 2010] Steinke, F., Hein, M., and Schölkopf, B. (2010). Nonparametric regression between general riemannian manifolds. *SIAM Journal on Imaging Sciences*, 3(3):527–563.

[Thanwerdas and Pennec, 2023] Thanwerdas, Y. and Pennec, X. (2023). O(n)-invariant riemannian metrics on spd matrices. *Linear Algebra and its Applications*, 661:163–201.

[Tuzel et al., 2006] Tuzel, O., Porikli, F., and Meer, P. (2006). Region covariance: A fast descriptor for detection and classification. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, pages 589–600, Berlin, Heidelberg. Springer Berlin Heidelberg.