UNIVERSITÄT TRIER

Riemannian optimization for advanced statistical models with clustered data

Vom Fachbereich IV der Universität Trier zur Verleihung des akademischen Grades

Doktorin der Naturwissenschaften (Dr. rer. nat.)

genehmigte

Dissertation

von

Lena Caroline Sembach

Trier, 2023

Berichterstattende:	PD Dr. Jan Pablo Burgard (Betreuer)
	Prof. Dr. Volker Schulz
	Assoc. Prof. Dr. Hiroyuki Sato
	Prof. Dr. Leonhard Frerick (Prüfungsvorsitz)
Datum der wissenschaftlichen Aussprache:	21. September 2022

Abstract

Modern decision making in the digital age is highly driven by the massive amount of data collected from different technologies and thus affects both individuals as well as economic businesses. The benefit of using these data and turning them into knowledge requires appropriate statistical models that describe the underlying observations well. Imposing a certain parametric statistical model goes along with the need of finding optimal parameters such that the model describes the data best. This often results in challenging mathematical optimization problems with respect to the model's parameters which potentially involve covariance matrices. Positive definiteness of covariance matrices is required for many advanced statistical models and these constraints must be imposed for standard Euclidean nonlinear optimization methods which often results in a high computational effort. As Riemannian optimization techniques proved efficient to handle difficult matrix-valued geometric constraints, we consider optimization over the manifold of positive definite matrices to estimate parameters of statistical models. The statistical models treated in this thesis assume that the underlying data sets used for parameter fitting have a clustering structure which results in complex optimization problems. This motivates to use the intrinsic geometric structure of the parameter space. In this thesis, we analyze the appropriateness of Riemannian optimization over the manifold of positive definite matrices on two advanced statistical models. We establish important problem-specific Riemannian characteristics of the two problems and demonstrate the importance of exploiting the Riemannian geometry of covariance matrices based on numerical studies.

Acknowledgements

I would like to express my sincere gratitude to my supervisors PD Dr. Jan Pablo Burgard and Prof. Dr. Volker Schulz who have jointly guided me during my PhD journey. I thank PD Dr. Jan Pablo Burgard for his guidance with statistical concepts, the interesting research ideas we developed together as well as his encouragement. I equally thank Prof. Dr. Volker Schulz for his guidance in numerical mathematics and for integrating me into his research group. His vast knowledge of optimization helped a lot in writing this thesis. Moreover, I would like to thank Associate Professor Dr. Hiroyuki Sato for accepting the position as a referee and thoroughly reading this thesis.

I owe thanks to the DFG Research Training Group 2126 ‘Algorithmic Optimization’ (ALOP), which employed me as a research assistant at Trier University during my PhD studies and which gave me the opportunity to attend several workshops and conferences.

I also thank all members of the research training group for many fruitful discussions and the pleasant working atmosphere. Especially, I would like to thank Maximilian Würschmidt and Manuel Klar for a lot of insightful discussions. I would also like to thank my officemate Britta Schmitt for the many fun moments we had together in E127 and elsewhere, she made the PhD time much more enjoyable. A special thanks also goes to Dr. Carina Moreira Costa, Luka Schlegel and Marc Harmening for proofreading parts of this thesis and giving me valuable feedback.

I am grateful for my parents Anni and Volker and my brother Philipp for their endless moral support throughout my academic studies and for always supporting me in every of my individual life decisions. I also thank my friends outside academia who have always been there for me, especially in times of doubts and exhaustion. Finally, I would like to thank Niklas for his patience, his ongoing encouragement and for his support in various directions.

Author's academic background

Lena Caroline Sembach

08/2019 – 09/2022	**Research assistant in the RTG ALOP** Universität Trier Prof. Dr. Volker Schulz
08/2017 – 07/2019	**Research assistant at Fraunhofer IESE** Kaiserslautern
06/2017	**Master of Science, Economathematics**
01/2014 – 06/2014	Semester abroad, Mathematics Université de Rennes 1, France
10/2013 – 06/2017	Master's program Economathematics at Karlsruher Institut für Technologie (KIT)
09/2013	**Bachelor of Science, Mathematics**
10/2010 – 09/2013	Bachelor's program Mathematics at Karlsruher Institut für Technologie (KIT)
04/2010	**Abitur**
08/2001 – 04/2010	Burggymnasium, Kaiserslautern

Preface

As part of this thesis, a refereed publication was written, which appears as [124] in the bibliography. The content of this work, including figures and results of numerical tests, can be found in Chapter 5.

Contents

CHAPTER 1

Introduction

In the last decades, the digital age has been shaping our society substantially. Due to the massive amount of information available, decision making in businesses and our individual life is based on data [121]. We can track our individual health status with wearables or predict defaults of machines with the help of connected technologies in the context of the *Industry of Things (IoT)* [121, 138]. The exponential growth of data drives the demand of mathematical models to support data-driven decision making. A famous quote in the context of the digital transformation is [126, 131]

> *"Information is the oil of the 21st century, and analytics is the combustion engine." Peter Sondergaard, 2011*

This underlines the importance of suitable models to gain insights from collected raw data, e.g. in the area of economics, health care and engineering. Data comes in many forms and shapes like images, time series, text. The choice for appropriate models to analyze observations is at the heart of data-driven decision making in the digital age [3].

Data-based decision making with statistical models assumes a specific mathematical model which typically incorporates parameters that need to be set such that the model describes the observed data well. Fitting such a parametric statistical model, that is finding appropriate values for the model parameters, typically results in solving an optimization problem [47, 105]. For a parameterized statistical model, the maximum likelihood approach is a common choice to address the fitting and answers the question [86]: *under the assumed statistical model, which parameters make the observed data most probable?*

In this thesis, we consider two different statistical models that are popular choices for data-driven decision making, namely *Gaussian mixture models* and *linear mixed models*. These models assume some clustering in the data. The former one, the Gaussian mixture model, makes the assumption that observations follow a probability distribution which is a mixture of K Gaussian distributions. This covers the typical situation where the data are assumed to be grouped into K clusters. In each cluster the data follow the same Gaussian distribution [3, Section 3.2.2]. Fitting a Gaussian mixture model consists of finding optimal parameters of the K Gaussian components as well as the mixing proportions, i.e. the proportion of each of the K components. The latter model, the linear mixed model, can be considered as a generalization of the classical linear model with clustered observations. It often is the model of choice when analyzing linear relationships if observations are collected from K different groups. The groups are assumed to capture a relevant part of the overall variance. In contrast to the Gaussian mixture model, the clustering is known beforehand and the structure is exploited during fitting [40, Section 1.17]. Linear mixed models are composed of *fixed effects*, like in a classical linear model, and *random effects* that model the variation introduced by considering specific clusters. Fitting a linear mixed model means to find optimal parameters of the effects parameters and to specify the distribution parameters of *within-group* and *between-group errors* [40, Section 1.1].

Beside the clustering assumption, the two models have another commonality being central to this thesis: the models' parameters incorporate covariance matrices. Covariance matrices are symmetric positive semidefinite [48]. For the models studied in this thesis, we assume they fulfill positive definiteness in order to guarantee well-definedness of the according objectives. This poses a challenge for classical nonlinear optimization algorithms since we need to optimize over the set of (symmetric) positive definite matrices which are open in the set of symmetric matrices of the same dimension. Hence, directly applying classical unconstrained nonlinear optimization algorithms carries the risk of leaving the feasible parameter space unless we explicitly impose positive definiteness. This can be done by incorporating additional constraints, for example by introducing a Cholesky decomposition. Yet, imposing such constraints usually comes with an unfavorable high computational effort [24]. As a surrogate, one can formulate the constraints as *geometric constraints*, that is to interpret the parameter space as a Riemannian manifold [2].

In the present thesis, we exploit the geometry of covariance matrices to impose positive definiteness by making use of the evolving field of Riemannian optimization [2, 120]. Riemannian optimization deals with the optimization of real-valued functions defined on a manifold, that is the consideration of the problem

$$\min_{\theta \in \mathcal{M}} f(\theta), \tag{1.1}$$

where θ is the parameter of interest and $\mathcal{M}$ is a Riemannian manifold. Optimization problems of the form (1.1) are constrained by parameters living on a Riemannian manifold. The basic idea of Riemannian optimization is to consider the problem (1.1) as an unconstrained problem and to generalize classical nonlinear unconstrained optimization theory to a manifold setting, i.e. to potentially nonlinear, curved spaces. Driven by many applications in engineering, physics and data mining, much research has been done in the last decades to build efficient algorithms for the optimization problem (1.1), see [1, 73, 120]. These algorithms can be used to solve important fitting problems of statistical models in a Riemannian framework and thus to address geometric constraints like a positive definiteness property of covariance matrices [17].

In this thesis, we use techniques from Riemannian optimization to solve optimization problems for two advanced statistical problems incorporating a clustering of data. To that end, we consider the manifold of positive definite matrices in order to model a positive definiteness constraint of covariance matrices. This allows us to exploit the intrinsic geometric structure of the parameter space yielding efficient solvers.

Contributions

The following contributions to the interplay of statistical analysis and mathematical optimization are achieved with this thesis:

Gaussian mixture models. For the fitting of Gaussian mixture models, a maximum likelihood approach is considered for the objective function. The contributions to Gaussian mixture models are based on the works by Hosseini and Sra [65, 66] who formulated the fitting problem as a Riemannian optimization problem and introduced important reformulations of the objective. In this thesis, we derive explicit expressions for the Riemannian gradient and the Riemannian Hessian for the problem of interest. This is a novelty compared to the works [65, 66]. The formula of the Riemannian Hessian allows to study second-order information of the problem and to use Newton-type methods for solving the Riemannian optimization problem. It contributes to a deeper understanding of the underlying geometry of Gaussian mixture models. Based on the attained problem-specific derivatives, we introduce a Riemannian Newton trust-region algorithm for Gaussian mixture models. We investigate the performance of the Riemannian Newton trust-region algorithm numerically. This includes implementation of the algorithm as well as numerical tests for various data settings. We provide numerical insights for both a clustering and a probability density estimation task with different levels of overlap of the K Gaussian components. Further, we compare the Riemannian Newton trust-region algorithm to other optimizers. These points bring algorithmic advances based on Riemannian geometry to the important class of Gaussian mixture models.

Linear mixed models. In this thesis, we focus on the task of estimating variance parameters of the linear mixed models, that is the covariance of the between-group error and the variance of the within-group error based on a residual maximum likelihood (REML) approach. In the course of this work, we formulate the problem of REML estimation as a Riemannian optimization problem which contributes to the geometric understanding of the parameter space of linear mixed models. To establish this reformulation, we take into account the specific structure of random effects covariance matrices and random effects design matrices. Based on the introduced Riemannian formulation of the optimization problem, we derive expressions for the Riemannian gradient and the Riemannian Hessian. These expressions are used for the application of a Riemannian Newton trust-region algorithm and a Riemannian nonlinear conjugate gradient method for linear mixed models. The establishment of these Riemannian algorithms for linear mixed models are used for the implementation as well as the numerical testing on different settings. We investigate the feasibility of Riemannian optimizers for linear mixed models in comparison to optimizers from the `lme4` package [44]. With this thesis, a novel geometric-based view on variance parameter estimation in linear mixed model is achieved. This is underlined by computational experiments giving rise to further investigation of important extensions of linear mixed models.

Structure of the thesis

The structure of this thesis is as follows. In **Chapter 2**, we introduce the foundations of Riemannian optimization. The chapter is divided into two major parts. The first part deals with the differential-geometric basics that we need to formally introduce optimization on Riemannian manifolds. We introduce the main concepts from an algorithmic rather than from an analytic perspective. This builds the foundation for the second part, where we generalize concepts from Euclidean nonlinear unconstrained optimization onto Riemannian manifolds. The chapter forms the basis for the establishment of geometric algorithms for the advanced statistical models studied in this work. One way to fit the statistical models is to use the approach of maximum likelihood estimation resulting in objectives that can be formulated as real-valued functions on Riemannian manifolds. **Chapter 3** is thus dedicated to the concept of maximum likelihood estimation. The statistical models studied can be considered as models with latent variables for what reason we study such a setting in Chapter 3 in further detail. We then review the Expectation Maximization algorithm which is designed for finding optimal parameters of parametric statistical models with latent variables. Fitting the statistical models of interest is based on a maximum likelihood approach in this thesis and involves the estimation of covariance matrices. The content of the subsequent chapter, **Chapter 4**, deals with the Riemannian geometry of covariance matrices and thus serves as a connection of the foundations of Riemannian optimization (Chapter 2) and maximum likelihood estimation (Chapter 3) with covariance matrices. We present the manifold of positive definite matrices as well as its Riemannian characteristics in order to formulate the maximum likelihood estimation problems studied in this thesis as Riemannian optimization problems. Equipped with an understanding of the underlying geometry of covariance matrices, we consider the aforementioned advanced statistical models in the subsequent chapters. In **Chapter 5**, we consider the problem of fitting Gaussian mixture models. We introduce the Riemannian optimization problem as established in the works [65, 66] and derive the Riemannian gradient and the Riemannian Hessian. Based on that, we develop a Riemannian Newton trust-region algorithm for Gaussian mixture models and analyze it both theoretically and numerically. The chapter is closed with a discussion of potential future work based on the present thesis. In **Chapter 6**, we consider variance estimation in linear mixed models with a Riemannian framework. We give an overview of linear mixed models and present the variance estimation problem for which we derive a Riemannian formulation of the associated optimization problem. Based on this novel geometric view on linear mixed models, we establish expressions for the according Riemannian gradient and Riemannian Hessian. We test the approach numerically by presenting computational results for two different Riemannian optimizers and discuss the results. Further, we give an overview of potential future research directions. The last chapter, **Chapter 7**, summarizes the findings of the research presented in this thesis.

CHAPTER 2

Foundations of Riemannian Optimization

Riemannian optimization has gained increasing interest in recent research due to its broad applicability to many problems in the area of engineering [82], computer vision [29], data science [140] and many others [120]. The field of Riemannian optimization addresses nonlinear objectives with geometric constraints, meaning that the solution of the optimization problem lies on a Riemannian manifold [2]. The authors of the popular text book by Absil, Baker and Gallivan [2] calls this "unconstrained optimization in a constrained search space". The basic idea of Riemannian optimization, also referred to as *manifold optimization* [22, 65], is to generalize concepts and algorithms of unconstrained Euclidean nonlinear optimization onto objectives defined on manifolds by considering local vector space approximations called *tangent spaces.* In this chapter, we give an introduction to Riemannian optimization which serves as the basis for the following chapters. This chapter explains the relevant concepts for Riemannian optimization rather from an algorithmic perspective than from an analytic perspective. We motivate differential-geometric concepts that are relevant for optimization and focus on these. For this, we follow the structure and notations from the textbooks [2, 22, 120]. For an analytic introduction to Riemannian manifolds, we refer to [80, 81]. In this chapter, we state all necessary theorems but omit their proofs and refer to according sources.

Due to the popularity of Riemannian optimization within the last years, e.g, in the area of engineering and data science, efforts on the development of software libraries has been expended. Toolboxes for Riemannian optimization include `Manopt` in Matlab [23], `pymanopt` in Python [136], `ROPTLIB` in C++ [68] and `Manopt.jl` in Julia [16].

The present chapter is divided into two major parts. In the first part, Section 2.1, we give an introduction to the differential-geometric framework needed for Riemannian optimization. This includes the formal definition of Riemannian manifolds as well as the formalization of higher-order information for functions defined on manifolds. A special focus is set on submanifolds embedded in the Euclidean space since these are of relevance for the statistical models studied in this thesis. In the second part of this chapter, Section 2.2, we generalize the concepts of unconstrained Euclidean nonlinear optimizations onto objectives defined on Riemannian manifolds. We present both first and second-order optimization algorithms for objectives defined on Riemannian manifolds which we later make use of in Chapter 5 and Chapter 6.

2.1 Elements of Riemannian Geometry

In this section, we introduce the differential-geometric definition of a manifold together with the necessary concepts to perform Riemannian optimization. In Section 2.1.1, we establish the definition of a (differentiable) manifold followed by the introduction of tangent spaces and differentiability of functions defined on manifolds in Section 2.1.2. Then, we generalize the notion of the gradient and Hessian onto the Riemannian setting in Section 2.1.3 and Section 2.1.4, respectively. Section 2.1.5 generalizes the understanding of shortest paths to manifolds. In this thesis, Euclidean submanifolds, that is subsets of Euclidean space with a special structure, are of particular interest. We introduce general embedded submanifolds in Section 2.1.6 and the special case of Euclidean submanifolds in Section 2.1.7.

2.1.1 Manifolds

Manifolds are spaces that locally resemble the Euclidean space, meaning that they can be locally identified with coordinate patches of $\mathbb{R}^d$ [21]. The identification with subsets in $\mathbb{R}^d$ is done via so-called *charts* and if the whole set can be endowed with suitable charts, it can be given a manifold structure. We establish the definition of a manifold formally in the following:

Definition 2.1. (chart) *[2, Section 3.1.1] A d-dimensional* chart *on a topological space M is a pair $(\mathcal{U}, \varphi)$ consisting of an open subset $\mathcal{U}$ of M and a homeomorphism $\varphi : \mathcal{U} \to V$ with $V \subset \mathbb{R}^d$ open.*

The concept of charts allows to study objects associated with $\mathcal{U}$ with real analysis by bringing them to their coordinate representation $\varphi(\mathcal{U})$ [2, Section 3.1.1]. For example, for a real-valued function $f : \mathcal{U} \to \mathbb{R}$, the *coordinate representative* $\tilde{f} = f \circ \varphi^{-1}$ is a function from $\mathbb{R}^d$ to $\mathbb{R}$ with domain $\varphi(\mathcal{U})$. This gives us an understanding of smoothness of a function defined on M: f is *smooth at* $\theta \in M$ if $\tilde{f}$ is smooth at $\varphi(\theta)$. Smoothness of f requires that two charts $(\mathcal{U}, \varphi)$, $(\mathcal{V}, \psi)$ on M yield the same conclusions regarding the differentiability of functions at θ: if a real-valued function f is defined on $\mathcal{U} \cap \mathcal{V}$, then $f \circ \varphi^{-1}$ and $f \circ \psi^{-1}$ should have the same differentiability properties on $\mathcal{U} \cap \mathcal{V}$ [22, Section 8.1]. *Compatibility* between charts captures this property:

Definition 2.2. (compatible charts) *[2, Section 3.1.1] Two charts $(\mathcal{U}, \varphi)$, $(\mathcal{V}, \psi)$ on M are* compatible *if they have the same dimension d and either $\mathcal{U} \cap \mathcal{V} = \emptyset$ or $\mathcal{U} \cap \mathcal{V} \neq \emptyset$ and*

1. *$\varphi(\mathcal{U} \cap \mathcal{V})$ is open in $\mathbb{R}^d$,*
2. *$\psi(\mathcal{U} \cap \mathcal{V})$ is open in $\mathbb{R}^d$ and*
3. *$\psi \circ \varphi^{-1} : \varphi(\mathcal{U} \cap \mathcal{V}) \to \psi(\mathcal{U} \cap \mathcal{V})$ is a diffeomorphism.*

To generalize the concept of differentiability to the whole set M, we must ensure that the whole set can be equipped with charts. Compatible charts that cover the whole set M form an atlas [22, Section 8.1]:

Definition 2.3. (atlas, maximal atlas) *[2, Section 3.1.1] A (d-dimensional)* atlas *$\mathcal{A}$ on a set M is a collection of pairwise compatible d-dimensional charts $(\mathcal{U}, \varphi)$ on M whose domains $\mathcal{U}$ cover M. For an atlas $\mathcal{A}$, the* maximal atlas *$\mathcal{A}^+$ is defined as the collection of charts on M which are compatible with all charts of $\mathcal{A}$. A maximal atlas is itself an atlas.*

The definition of a maximal atlas allows us to equip a set M with a differentiable structure and to have a meaningful representation of locally linearizable spaces. However, we need to impose additional topological assumptions on the atlas to study optimization theory on such spaces. First, we need to avoid that a sequence of points can converge to more than one limit point. Second, we need a notion of length via a *Riemannian metric* [22, Section 8.2]. This yields the following definition of a manifold which we use throughout this thesis.

Definition 2.4. (manifold) *[2, Section 3.1.1] A d-dimensional* differentiable manifold *is a tuple $\mathcal{M} = (M, \mathcal{A}^+)$, where M is a set and $\mathcal{A}^+$ is a d-dimensional maximal atlas such that the topology induced by $\mathcal{A}^+$ is Hausdorff and second-countable.*

A few remarks to the Definition 2.4 are in order:

(i) Two techniqual requirements are made in Definition 2.4: the atlas topology is assumed Hausdorff, i.e. any two disjoint points of M have disjoint neighborhoods, and assumed to be second-countable, i.e. the atlas topology has a countable basis (for formal definitions, see Appendix A.1). These assumptions ensure that the considerations above hold true: with Definition 2.4, any convergent sequence in M has a unique limit point in M and the manifold admits a Riemannian metric. For details, see [81].

(ii) In the literature, a manifold is sometimes simply defined as a set endowed with a maximal atlas, which does not preclude certain counterintuitive properties of $(M, \mathcal{A}^+)$ (see e.g. [120, Section 2.3] for such examples). Throughout this thesis, we stick to Definition 2.4 when we talk about manifolds. Further, when writing "manifolds" in this thesis, we always mean differentiable manifolds in the sense of Definition 2.4.

(iii) If $\mathcal{M} = (M, \mathcal{A}^+)$ is a manifold, we will usually drop the atlas and use $\mathcal{M}$ and M interchangeably when there is no ambiguity.

(iv) When talking about manifolds $\mathcal{M}$ in this thesis, we assume they are *connected*, meaning that $\mathcal{M}$ cannot be expressed as the disjoint union of two nonempty sets. This ensures that we can find a path contained in $\mathcal{M}$ that connects any two points on the manifold [2, Section 3.1.1].

In practice, it is necessary to check the Hausdorff and second-countable property of an atlas topology. The following result provides a helpful characterization of an atlas to fulfill these properties:

Theorem 2.5. *[22, Proposition 8.22] Let $\mathcal{A}$ be an atlas for the set M. Assume that*

1. *For all $\theta_1, \theta_2 \in M$ distinct, either both θ_1 and θ_2 are in the domain of some chart, or there exist two disjoint chart domains $\mathcal{U}, \mathcal{V}$ such that $\theta_1 \in \mathcal{U}$ and $\theta_2 \in \mathcal{V}$ and*

2. *Countably many of the chart domains suffice to cover M.*

Then the atlas topology of $\mathcal{A}^+$ is Hausdorff and second-countable, so that $\mathcal{M} = (M, \mathcal{A}^+)$ is a manifold.

Proof. For a proof of Theorem 2.5, see [81, Lemma 1.35]. □

Coordinate representations

With the definition of a manifold, we are able to consider differentiability of functions F mapping from a manifold $\mathcal{M}_1$ of dimension d_1 to a manifold $\mathcal{M}_2$ of dimension d_2 [2, Section 3.2]. For $\theta \in \mathcal{M}_1$, we choose charts φ_1, φ_2 around θ and $F(\theta) \in \mathcal{M}_2$, respectively. We introduce the *coordinate representation* $\hat{F}$ of F given by

$$\hat{F} = \varphi_2 \circ F \circ \varphi_1^{-1} : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}. \tag{2.1}$$

Then, we say that F is *differentiable* or *smooth at* θ if $\hat{F}$ is of class C^∞ at $\varphi_1(\theta)$. Further, a function $F : \mathcal{M}_1 \to \mathcal{M}_2$ is called *smooth* if it is smooth at every point of its domain [2, Section 3.2].

Product manifolds

In many applications, variables of interest for optimization live on different manifolds which together again constitute a manifold, namely a *product manifold* [2, Section 3.1.6]. Product manifolds are composed of several single manifolds. Formally, let $\mathcal{M}_1, \mathcal{M}_2$ be manifolds of dimensions d_1, d_2, respectively. Let $(\mathcal{U}_1, \varphi_1)$ and $(\mathcal{U}_2, \varphi_2)$ be charts of the manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively and $\theta = (\theta_1, \theta_2) \in M_1 \times M_2$ of dimension $d_1 + d_2$, where $\theta_1 \in M_1$, $\theta_2 \in M_2$. Then, the mapping

$$\varphi_1 \times \varphi_2 : \mathcal{U}_1 \times \mathcal{U}_2 \to \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : (\theta_1, \theta_2) \mapsto (\varphi_1(\theta_1), \varphi_2(\theta_2))$$

is a chart for the set $M_1 \times M_2$. We can thus form an atlas for the set $M_1 \times M_2$ out of the charts of $\mathcal{M}_1$ and $\mathcal{M}_2$ and endow $M_1 \times M_2$ with a differentiable structure. The resulting manifold $\mathcal{M}_1 \times \mathcal{M}_2$ is called the *product* of the manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$ and has dimension $d_1 + d_2$. Its manifold topology is equivalent to the product topology [2, Section 3.1.6].

2.1.2 Tangent Spaces and Differential Maps

With the definition of a manifold (Definition 2.4), we are able to generalize real analysis to manifolds. To study differentiability of a function $f : \mathcal{M} \to \mathbb{R}$ at a point θ without considering their coordinate representations, we introduce *tangent vectors* and *tangent spaces* formally.

Definition 2.6. (curve, tangent vector, tangent space, tangent bundle, vector field)*[2, Section 3.5.1] Let $\mathcal{M}$ be a manifold.*

1. *A smooth mapping $\gamma \in C^\infty$ with*
$$\gamma : I \to \mathcal{M} : t \mapsto \gamma(t), \quad 0 \in I \subseteq \mathbb{R}$$
is called a curve *in $\mathcal{M}$.*

2. *Let $\mathfrak{F}_\theta(\mathcal{M})$ denote the set of all smooth, real-valued functions defined on a neighbourhood of θ. A* tangent vector *ξ_θ at $\theta \in \mathcal{M}$ is a mapping from $\mathfrak{F}_\theta(\mathcal{M})$ to $\mathbb{R}$ such that there exists a curve γ on $\mathcal{M}$ with*
$$\begin{aligned} &\gamma(0) = \theta, \\ &\xi_\theta f = \dot{\gamma}(0) f \coloneqq \frac{\mathrm{D}}{\mathrm{d}\, t} f(\gamma(t))|_{t=0} \quad \forall f \in \mathfrak{F}_\theta(\mathcal{M}). \end{aligned}$$
Such a curve is said to realize *the tangent vector ξ_θ. The point θ is called the* foot *of the tangent vector ξ_θ.*

3. *The* tangent space *to $\mathcal{M}$ at a point θ, denoted by $T_\theta\mathcal{M}$, is the set of all tangent vectors to $\mathcal{M}$ at θ.*

4. *The* tangent bundle *$T\mathcal{M}$ is defined as the union of tangent spaces at all points from $\mathcal{M}$, i.e.*
$$T\mathcal{M} \coloneqq \bigcup_{\theta \in \mathcal{M}} T_\theta\mathcal{M} = \{(\theta, \xi) : \theta \in \mathcal{M}, \xi \in T_\theta\mathcal{M}\}.$$

5. *A* vector field *V on a manifold $\mathcal{M}$ is a smooth function from $\mathcal{M}$ to the tangent bundle $T\mathcal{M}$ that assigns to each point $\theta \in \mathcal{M}$ a tangent vector in $T_\theta\mathcal{M}$.*

Tangent spaces are an important tool for Riemannian optimization technique as they give local vector space approximations around each point of a manifold. Like this, we do not necessarily need to specify the atlas of a manifold to apply concepts of real analysis [2, Section 3.5].

We make a few remarks for tangent vectors and spaces:

(i) Tangent spaces are vector spaces: let $\dot{\gamma}_1(0), \dot{\gamma}_2(0)$ in $T_\theta\mathcal{M}$ and $\alpha, \beta \in \mathbb{R}$. Define
$$(\alpha\dot{\gamma}_1(0) + \beta\dot{\gamma}_2(0)) f \coloneqq \alpha(\dot{\gamma}_1(0) f) + \beta(\dot{\gamma}_2(0) f).$$
Consider a chart $(\mathcal{U}, \varphi)$ with $\theta \in \mathcal{U}$ and the curve $\gamma(t) = \varphi^{-1}(\alpha\varphi(\gamma_1(t)) + \beta\varphi(\gamma_2(t)))$. Then we see that $\dot{\gamma}(0) = \alpha\dot{\gamma}_1(0) + \beta\dot{\gamma}_2(0)$ yielding that $\alpha\dot{\gamma}_1(0) + \beta\dot{\gamma}_2(0)$ is a well-defined tangent vector at θ [2, Section 3.5.1].

(ii) The tangent vector at a point θ is well-defined, i.e. the definition does not depend on the choice of the curve γ, see [2, Section 3.5.4] for details.

(iii) We sometimes drop the index and simply write ξ for a tangent vector when there is no ambiguity to which tangent space it refers.

With the definition of tangent vectors, we are now in a position where we can define derivatives on manifolds without using the coordinate representation (2.1) [2, Section 3.5.6]. For this, let $F : \mathcal{M}_1 \to \mathcal{M}_2$ be a smooth mapping between two manifolds $\mathcal{M}_1$ and $\mathcal{M}_2$ and ξ_θ be a tangent vector at a point θ of $\mathcal{M}_2$. Then, the mapping $\mathrm{D}\,F(\theta)[\xi_\theta]$ from $\mathfrak{F}_{F(\theta)}(\mathcal{M}_2)$ to $\mathbb{R}$ defined by

$$(\mathrm{D}\,F(\theta)[\xi_\theta])f \coloneqq \xi_\theta(f \circ F)$$

is a tangent vector to $\mathcal{M}_2$ at $F(\theta)$. The tangent vector $\mathrm{D}\,F(\theta)[\xi_\theta]$ is realized by $F \circ \gamma$, where γ is any curve that realizes ξ_θ. The mapping

$$\mathrm{D}\,F(\theta) : T_\theta\mathcal{M}_1 \to T_{F(\theta)}\mathcal{M}_2 : \xi_\theta \mapsto \mathrm{D}\,F(\theta)[\xi_\theta]$$

is a linear mapping called the *differential* (or *differential map, derivative* or *tangent map*) of F at θ [2, Section 3.5.6].

2.1.3 Riemannian Metrics and Gradients

In the previous sections, we introduced manifolds with local linear space approximations. Endowing the local vector space approximations with inner products provides a notion of length and angles on these spaces [2, Section 3.6]. We formally define *Riemannian metrics* in the following.

Definition 2.7. (Riemannian metric)*[120, Definition 3.14] Let $\mathcal{M}$ be a differentiable manifold. For any $\theta \in \mathcal{M}$, let $g_\theta : T_\theta\mathcal{M} \times T_\theta\mathcal{M} \to \mathbb{R}$ be an inner product on $T_\theta\mathcal{M}$. If $g : \theta \mapsto g_\theta$ is smooth, g is called a* Riemannian metric *on $\mathcal{M}$ and the pair $(\mathcal{M}, g)$ is called a* Riemannian manifold.

A few remarks for Definition 2.7 are in order:

(i) Throughout this thesis, we will often refer to a Riemannian manifold $(\mathcal{M}, g)$ simply as $\mathcal{M}$ when the metric is clear from the context.

(ii) Since g_θ is an inner product on $T_\theta\mathcal{M}$, we mostly use the notation $\langle \cdot, \cdot \rangle_\theta$ for g_θ.

(iii) We denote the *Riemannian norm* at θ by $\|\xi\|_\theta := \sqrt{g_\theta(\xi_\theta, \chi_\theta)} = \sqrt{\langle \xi, \xi \rangle_\theta}$ for a tangent vector $\xi \in T_\theta\mathcal{M}$.

(iv) One can show that for every manifold in the sense of Definition 2.4, there exists a Riemannian metric (see [120, Theorem 3.3] and Appendix A.1).

(v) When working with product manifolds, one can easily construct a Riemannian metric out of the Riemannian metrics of the single components:
Let $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_K$ be a product manifold and $\theta = (\theta_1, \dots, \theta_K) \in \mathcal{M}$, $\theta_j \in \mathcal{M}_j$ and $\xi_\theta = (\xi_{\theta_1}, \dots, \xi_{\theta_K})$, $\chi_\theta = (\chi_{\theta_1}, \dots, \chi_{\theta_K})$ corresponding tangent vectors, that is $\xi_{\theta_j}, \chi_{\theta_j} \in T_{\theta_j}\mathcal{M}_j$. Further, let $\langle \cdot, \cdot \rangle_{\theta_j}$ be inner products on the tangent spaces $T_{\theta_j}\mathcal{M}_j$ that vary smoothly with θ_j in the sense of Definition 2.7. Then,

$$\langle \xi_\theta, \chi_\theta \rangle_\theta = \langle \xi_{\theta_1}, \chi_{\theta_1} \rangle_{\theta_1} + \cdots + \langle \xi_{\theta_K}, \chi_{\theta_K} \rangle_{\theta_K} \tag{2.2}$$

defines an inner product on $T_\theta\mathcal{M}$ that varies smoothly with θ, making $\mathcal{M}$ a Riemannian manifold [22, Section 3.7].

For a function $f : \mathcal{M} \to \mathbb{R}$, we seek to define the *gradient*, which is a vector field on $\mathcal{M}$. On Riemannian manifolds, we can generalize the Euclidean gradient to the manifold setting by identifying it with the differential and the inner product:

Definition 2.8. (Riemannian gradient) *[2, Section 3.5] Let $\mathcal{M}$ be a Riemannian manifold and $f : \mathcal{M} \to \mathbb{R}$ smooth. The* Riemannian gradient *of f at θ, denoted by $\operatorname{grad} f(\theta)$, is defined as the unique element of $T_\theta\mathcal{M}$ that satisfies*

$$\langle \operatorname{grad} f(\theta), \xi_\theta \rangle_\theta = \mathrm{D}\, f(\theta)[\xi_\theta] \quad \forall \xi_\theta \in T_\theta\mathcal{M},$$

where $\langle \cdot, \cdot \rangle_\theta$ is the inner product defined on $T_\theta\mathcal{M}$.

When $\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_K$ is a product manifold, one can easily see with the inner product (2.2) that the gradient $\operatorname{grad} f(\theta) \in T_\theta\mathcal{M}$ reads

$$\operatorname{grad} f(\theta) = \left(\operatorname{grad}_{\theta_1} f(\theta), \dots, \operatorname{grad}_{\theta_K} f(\theta)\right),$$

where $\operatorname{grad}_{\theta_j} f(\theta) \in T_{\theta_j}\mathcal{M}_j$ is the Riemannian gradient of f with respect to θ_j [22, Section 3.8].

2.1.4 Affine Connections and the Riemannian Hessian

In the last section, we introduced the Riemannian gradient which characterizes first derivatives on Riemannian manifolds. In this section, we seek to generalize the second derivative for Riemannian manifolds. To do so, we note that differentiating the gradient means differentiating over vector fields, for what reason we introduce *Riemannian connections*, which are *affine connections* with special characteristics.

Definition 2.9. (affine connection, covariant derivative) *[2, Section 5.2] Let $\mathcal{M}$ be a manifold. We denote the set of all vector fields on $\mathcal{M}$ by $\mathfrak{X}(\mathcal{M})$. We define:*

1. *Given a vector field V on $\mathcal{M}$ and a smooth real-valued function f on $\mathcal{M}$, let Vf denote the function on $\mathcal{M}$ defined by*

$$(Vf)(\theta) := \xi_\theta f,$$

for all $\theta \in \mathcal{M}$ and $\xi_\theta f$ as in Definition 2.6.

2. *An* affine connection ∇ *on a manifold is a mapping*

$$\nabla : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M}) : (V, U) \mapsto \nabla_V U$$

which satisfies the following properties:

   (a) *$\mathfrak{F}(\mathcal{M})$-linearity in V: $\nabla_{f_1 V + f_2 W} U = f_1 \nabla_V U + f_2 \nabla_W U$,*

   (b) *$\mathbb{R}$-linearity in U: $\nabla_V(\alpha U + \beta \tilde{U}) = \alpha \nabla_V U + \beta \nabla_V \tilde{U}$,*

   (c) *Leibniz' law: $\nabla_V(fU) = (Vf)U + f\nabla_V U$,*

for any $U, \tilde{U}, V, W \in \mathfrak{X}(\mathcal{M})$, $f, f_1, f_2 \in \mathfrak{F}(\mathcal{M})$ and $\alpha, \beta \in \mathbb{R}$. The vector field $\nabla_V U$ is called the covariant derivative *of U with respect to V for the affine connection ∇.*

3. *An* affine connection at θ *is given by restricting an affine connection to θ, i.e. it is the mapping*

$$T_\theta \mathcal{M} \times \mathfrak{X}(\theta) \to \mathfrak{X}(\theta) : (\nabla_{V_\theta}, \xi_\theta) \mapsto \nabla_{V_\theta} \xi_\theta,$$

where $\mathfrak{X}(\theta)$ denotes the set of vector fields on $\mathcal{M}$ whose domain includes θ.

With affine connections, we are able to differentiate vector fields like the Riemannian gradient along other vector fields. It can be shown that there exists at least one affine connection for every manifold [2, Proposition 5.2.1]. However, uniqueness is not necessarily given and we need to impose additional properties on the affine connection in order to achieve uniqueness. To formalize these properties, we first introduce some helpful notations [22, Definition 5.5]:

For $U, V \in \mathfrak{X}(\mathcal{M})$ and $f \in \mathfrak{F}(\mathcal{U})$ for $\mathcal{U}$ open in $\mathcal{M}$, let

i) $Uf \in \mathfrak{F}(\mathcal{U})$ such that $(Uf)(\theta) = \mathrm{D}\, f(\theta)[U(\theta)]$,

ii) $[U, V] : \mathfrak{F}(\mathcal{U}) \to \mathfrak{F}(\mathcal{U})$, $f \mapsto U(Vf) - V(Uf)$ (*Lie-bracket* of U and V)

iii) $\langle U, V\rangle \in \mathfrak{F}(\mathcal{M})$ such that $\langle U, V\rangle(\theta) = \langle U(\theta), V(\theta)\rangle_\theta$

With these notations, we can state the following important theorem which gives us uniqueness of a specific affine connection, the *Levi-Civita connection*.

Theorem 2.10. *(Levi-Civita, [2, Theorem 5.2.1]) On a Riemannian manifold $\mathcal{M}$, there exists a unique affine connection ∇ that satisfies*

i) Symmetry: $\nabla_V U - \nabla_U V = [U, V]$ and

ii) Compatibility with the Riemannian metric: $W\langle U, V\rangle = \langle \nabla_W U, V\rangle + \langle U, \nabla_W U\rangle$.

The affine connection satisfying i) and ii) is called the Levi-Civita connection *or the* Riemannian connection *of $\mathcal{M}$. It is the unique solution to the* Koszul-formula

$$2\langle \nabla_W V, U\rangle = W\langle V, U\rangle + V\langle U, W\rangle - U\langle W, V\rangle - \langle W, [U, V]\rangle + \langle V, [U, W]\rangle + \langle U, [V, W]\rangle. \tag{2.3}$$

Proof. We refer to [22, Section 5.4 & 9.10] for a proof. □

With the Riemannian connection, we are able to generalize the *Hessian* to functions defined on manifolds.

Definition 2.11. (Riemannian Hessian)*[2, Section 5.5] Let f be a real-valued smooth function on a Riemannian manifold $\mathcal{M}$ and ∇ the according Riemannian connection. Then the* Riemannian Hessian *of f at a point θ in $\mathcal{M}$ is the linear mapping*

$$\operatorname{Hess} f(\theta) : T_\theta\mathcal{M} \to T_\theta\mathcal{M}$$

defined by

$$\operatorname{Hess} f(\theta)[\xi_\theta] = \nabla_{\xi_\theta} \operatorname{grad} f(\theta) \tag{2.4}$$

for all ξ_θ in $T_\theta\mathcal{M}$.

Hence, at any point $\theta \in \mathcal{M}$, the Riemannian Hessian defines a linear operator from the tangent space $T_\theta\mathcal{M}$ onto itself. The operator $\operatorname{Hess} f$ maps elements from $\mathfrak{X}(\mathcal{M})$ to elements from $\mathfrak{X}(\mathcal{M})$ as $\operatorname{Hess} f[V] = \nabla_V \operatorname{grad} f$.

A few remarks are in order:

(i) The Riemannian Hessian is self-adjoint with respect to the Riemannian metric, that is for all $\theta \in \mathcal{M}$ and $\xi, \chi \in T_\theta\mathcal{M}$ [22, Proposition 8.66],

$$\langle \operatorname{Hess}(f(\theta)[\xi]), \chi\rangle_\theta = \langle \xi, \operatorname{Hess} f(\theta)[\chi]\rangle_\theta.$$

(ii) The definition of the Riemannian Hessian by (2.4) ensures that $\operatorname{Hess} f(\theta)[\xi_\theta]$ is an element of $T_\theta\mathcal{M}$. This is not ensured if we simply take the derivative of gradient vector fields, for an example see [22, Section 5.1].

(iii) When $\mathcal{M} = \bigtimes_{j=1}^{K} \mathcal{M}_j$ is a product manifold and $\nabla^{(1)}, \dots, \nabla^{(K)}$ are the respective Riemannian connections of $\mathcal{M}_1, \dots, \mathcal{M}_K$, one can show that the single components of the Riemannian Hessian

$$\operatorname{Hess} f(\theta)[\xi_\theta] = \Big((\operatorname{Hess} f(\theta))_{\theta_1}, \dots, (\operatorname{Hess} f(\theta))_{\theta_K} \Big)$$

are given by

$$(\operatorname{Hess} f(\theta))_{\theta_j} = \nabla^{(j)}_{\xi_{\theta_j}} \operatorname{grad}_{\theta_j} f(\theta_j) + \sum_{\substack{r=1 \\ r\neq j}}^{K} \mathrm{D}_{\theta_r} \Big(\operatorname{grad}_{\theta_j} f(\theta_r) \Big) [\xi_{\theta_r}], \tag{2.5}$$

see [22, Sections 5.3 & 5.4]. Here, the expression $\operatorname{grad}_{\theta_j} f(\theta_r)$ denotes the gradient at position θ_j, where all other variables than θ_r are assumed fixed (differentiation with respect to θ_r).

2.1.5 Generalizing Straight Lines: Geodesics and Distances

With the concept of Riemannian connections, we are able to generalize the concept of straight lines in $\mathbb{R}^d$ to manifolds. A geometric interpretation of straight lines $t \mapsto x + tv$ for $x, v \in \mathbb{R}^d$ is given by their vanishing second derivative, i.e. they have zero *acceleration* [22, Section 10.1]. We have already introduced the notion of a tangent vector $\dot\gamma$ for a curve γ in a manifold $\mathcal{M}$, yielding an interpretation as the *velocity* of the curve γ at t. The according mapping $t \mapsto \dot\gamma(t)$ defines the *velocity vector field* along γ [2, Section 5.4]. We define the *acceleration vector field* $\ddot\gamma$ of the curve γ in the following. To that end, let $\mathcal{M}$ be a manifold equipped with an affine connection ∇ and let γ be a curve in $\mathcal{M}$ with domain $I \subseteq \mathbb{R}$. A *smooth vector field on the curve* γ smoothly assigns to each $t \in I$ a tangent vector to $\mathcal{M}$ at $\gamma(t)$. The set of all smooth vector fields on γ is denoted by $\mathfrak{X}(\gamma)$. It can be shown [80, Chapter 4] that there is a unique function $\xi \mapsto \frac{\mathrm{D}}{\mathrm{d}t}\xi$ from $\mathfrak{X}(\gamma)$ to $\mathfrak{X}(\gamma)$ such that

1. $\frac{\mathrm{D}}{\mathrm{d}\,t}(a\xi + b\zeta) = a\frac{\mathrm{D}}{\mathrm{d}\,t}\xi + b\frac{\mathrm{D}}{\mathrm{d}\,t}\zeta \quad (a, b \in \mathbb{R})$,
2. $\frac{\mathrm{D}}{\mathrm{d}\,t}(f\xi) = f'\xi + f\frac{\mathrm{D}}{\mathrm{d}\,t}\xi \quad (f \in \mathfrak{F}(I))$,
3. $\frac{\mathrm{D}}{\mathrm{d}\,t}(\eta \circ \gamma)(t) = \nabla_{\dot{\gamma}(t)}\eta \quad (t \in I, \eta \in \mathfrak{X}(\mathcal{M}))$.

Based on that, we define the *acceleration vector field*:

Definition 2.12. (acceleration vector field)*[2, Section 5.4] The* acceleration vector field $\frac{\mathrm{D}^2}{\mathrm{d}\,t^2}\gamma$ *on* γ *is given by*

$$\ddot{\gamma} = \frac{\mathrm{D}^2}{\mathrm{d}\,t^2}\gamma := \frac{\mathrm{D}}{\mathrm{d}\,t}\dot{\gamma}.$$

With this definition, we can generalize straight lines in $\mathbb{R}^d$ to a manifold via the zero acceleration property:

Definition 2.13. (geodesic)*[2, Section 5.4] A* geodesic γ *on a manifold* $\mathcal{M}$ *endowed with an affine connection* ∇ *is a curve with zero acceleration, i.e.*

$$\ddot{\gamma}(t) = 0$$

for all t *in the domain of* γ*.*

We note that different affine connections produce different geodesics, see [2, Section 5.4] for details.

Another interpretation of straight lines is to think about them as the shortest path between two points in $\mathbb{R}^d$. We need a notion of *length* to study a similar characterization for manifolds:

Definition 2.14. (length of a curve, Riemannian distance, minimizing curve)*[2, Section 3.6] Let* $(\mathcal{M}, g)$ *be a Riemannian manifold.*

i) The length of a curve $\gamma : \mathbb{R} \supseteq I \to \mathcal{M}$ *is given by*

$$L(\gamma) = \int_I \sqrt{g(\dot{\gamma}(t), \dot{\gamma}(t))}\,\mathrm{d}\,t.$$

ii) The Riemannian distance *is given by*

$$\operatorname{dist} : \mathcal{M} \times \mathcal{M} \to \mathbb{R} : (\theta_1, \theta_2) \mapsto \inf_{\gamma \in \Gamma_{\theta_1,\theta_2}} L(\gamma), \tag{2.6}$$

where $\Gamma_{\theta_1,\theta_2}$ *is the set of all curves in* $\mathcal{M}$ *joining the points* θ_1*,* θ_2 *in* $\mathcal{M}$*.*

iii) If the infimum in (2.6) *is attained for some curve γ, we call γ a* minimizing curve*.*

One can show that, up to parameterization, minimizing curves are geodesics. For details, see [80, Chapter 6]. This result shows that there is a clear connection between the two viewpoints of straight lines: the generalization of the zero-acceleration characteristic and the generalization of the shortest-path property [22, Section 10.1].

2.1.6 Submanifolds

An important special case of manifolds are submanifolds, i.e. subsets of manifolds with a suitable differential structure. For this, we consider a manifold $\mathcal{M} = (M, \mathcal{A}^+)$ and a subset $N \subset M$. We first study the special case where N is an open subset of M. For any chart $(\mathcal{U}, \varphi)$ of $\mathcal{M}$ with $\mathcal{U} \cap N \neq \emptyset$, we build the chart $(\mathcal{U} \cap N, \varphi)$ on N. The collection of these charts naturally forms a (maximal) atlas $\mathcal{B}^+$ for N, turning it into an atlas for N and turning it into a manifold [22, Section 8.2]. The manifold $\mathcal{N} = (N, \mathcal{B}^+)$ is called an *open submanifold* of $\mathcal{M}$ [22, Definition 8.24].

For more general subsets N of M, we consider *immersed submanifolds* with a special topology. In order to define these, we introduce the *rank* of a differentiable function F mapping from a manifold $\mathcal{M}_1$ of dimension d_1 to another manifold $\mathcal{M}_2$ of dimension d_2. Given a point $\theta \in \mathcal{M}_1$, the *rank* of F is the dimension of the range of the Jacobian $\mathrm{D}\,\hat{F}(\varphi_1(\theta))[\cdot] : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$, where $\hat{F}$ is a coordinate representation of F given by (2.1) and $\mathrm{D}\,\hat{F}(\varphi_1(\theta))$ denotes the (Fréchet) differential of $\hat{F}$ [2, Section 3.2.1].

With this notation, we define *immersed submanifolds* and *embedded submanifolds*:

Definition 2.15. (immersed submanifold, embedded submanifold)*[2, Section 3.1.1] Consider two manifolds $\mathcal{N} = (N, \mathcal{B}^+)$, $\mathcal{M} = (M, \mathcal{A}^+)$ where $N \subset M$. We introduce the* inclusion map

$$i : N \to M : \theta \mapsto \theta$$

which maps points in N to themselves in M. Then we define

1. *If i is smooth and $\mathrm{D}\,i(\theta)$ has rank equal to* $\dim \mathcal{N}$ *for all $\theta \in N$, we say that $\mathcal{N}$ is an* immersed submanifold *of $\mathcal{M}$.*

2. *If $\mathcal{N}$ is an immersed submanifold of $\mathcal{M}$ and its atlas topology coincides with the subspace topology of $N \subset M$ induced from the topological space $\mathcal{M}$ (i.e. every open set of N is the intersection of some open set of M with N), then $\mathcal{N}$ is called an* embedded submanifold of $\mathcal{M}$*, while $\mathcal{M}$ is called the* embedding space *of $\mathcal{N}$.*

Whereas immersed submanifolds give some freedom regarding the differentiable structure of the manifold, the notion of an embedded submanifold requires a specific one that is compatible with $\mathcal{M}$. It can be shown that given a subset N of a manifold $\mathcal{M}$, there exists at most one such compatible structure:

Theorem 2.16. *[81, Theorem 5.13] Let $\mathcal{M} = (M, \mathcal{A}^+)$ be a manifold. A subset N of M admits at most one differentiable structure that makes it an embedded submanifold of $\mathcal{M}$.*

Proof. We refer to [81, Chapter 5] for a proof. □

With this result, we can give a complete characterization of (embedded) submanifolds:

Theorem 2.17. *[22, Theorem 8.75] Let $\mathcal{M} = (M, \mathcal{A}^+)$ be a manifold. A subset $N \neq \emptyset$ of M is an embedded submanifold of $\mathcal{M}$ if either of the following holds:*

1. *N is an open subset of $\mathcal{M}$. Then, we call $\mathcal{N} = (N, \mathcal{B}^+)$ an* open submanifold *of $\mathcal{M}$ and $\dim \mathcal{N} = \dim \mathcal{M}$.*

2. *For a fixed integer $d \geq 1$ and for each $\theta \in N$ there exists a smooth neighborhood $\mathcal{V}$ of θ in $\mathcal{M}$ and a smooth function $h : \mathcal{V} \to \mathbb{R}^d$ such that*

   a) if $\bar{\theta}$ is in $\mathcal{V}$, then $\bar{\theta} \in N$ if and only if $h(\bar{\theta}) = 0$; and

   b) $\operatorname{rank} \mathrm{D}\, h(\bar{\theta}) = d$ for all $\bar{\theta} \in \mathbb{N} \cap \mathcal{V}$.

   Then, $\dim \mathcal{N} = \dim \mathcal{M} - d$ and h is called a local defining function.

Proof. A proof can be found in [81, Chapter 5]. □

2.1.7 Submanifolds of Euclidean Spaces

In this thesis, we will consider submanifolds of Euclidean spaces. As some of the concepts introduced in the previous sections simplify for this special case, we study *Euclidean submanifolds* in more detail. We formalize Euclidean submanifolds and state the concepts of the previous sections for Euclidean submanifolds in the following.

Let $\mathcal{E}$ denote a vector space over the reals, that is a set equipped with (and closed under) vector addition and scalar multiplication by real numbers. Typical examples include $\mathbb{R}^d$, $\mathbb{R}^{d_1 \times d_2}$ or the set of real, symmetric matrices of size d denoted by $\mathbb{S}^d$. A vector space $\mathcal{E}$ equipped with an inner product is an *Euclidean space* [22, Section 3.1]. Since an inner product can be defined for each vector space over the reals $\mathcal{E}$, we will use the notation $\mathcal{E}$ also for an Euclidean space. Euclidean spaces have a natural linear manifold structure, which becomes clear when we consider the set of real matrices of dimension $d_1 \times d_2$ denoted by $\mathbb{R}^{d_1 \times d_2}$ as a representative example [2, Section 3.1.5]. Consider the mapping $\varphi : \mathbb{R}^{d_1 \times d_2} \to \mathbb{R}^{d_1 d_2} : A \mapsto \operatorname{vec}(A)$, where $\operatorname{vec}(A)$ denotes the vector obtained by stacking the columns of A below one another. An atlas for $\mathbb{R}^{d_1 \times d_2}$ can be formed by this single chart, and by Theorem 2.5, $\mathcal{E}$ admits a manifold structure. Accordingly, we can build a manifold structure for any Euclidean space $\mathcal{E}$ by using such a vectorization [22, Section 7.1].

In this thesis, embedded submanifolds of $\mathcal{E}$ are of particular interest. Subsets of the manifold $\mathcal{E}$ can be given differentiable structures as pointed out in Section 2.1.6. With Theorem 2.17, we are able to define embedded submanifolds in $\mathcal{E}$ as *Euclidean submanifolds* in the following way:

Definition 2.18. (Euclidean submanifold) *[22, Definition 3.10] Let $\mathcal{E}$ be a linear space of dimension d. A subset $\mathcal{M}$ of $\mathcal{E}$ is an* embedded submanifold of $\mathcal{E}$ *or an* Euclidean submanifold *of dimension n if either of the following holds:*

1. *$n = d$ and $\mathcal{M}$ is open in $\mathcal{E}$. Then, we call $\mathcal{M}$ an* open submanifold. *If $\mathcal{M} = \mathcal{E}$, we call it a* linear manifold*,*
2. *$n = d - k$ for some $k \geq 1$ and, for each $\theta \in \mathcal{M}$ there exists a neighborhood $\mathcal{U}$ of θ in $\mathcal{E}$ and a smooth function $h : \mathcal{U} \to \mathbb{R}^k$ such that*
   a) if $\bar{\theta}$ is in $\mathcal{U}$, then $\bar{\theta} \in \mathcal{M}$ if and only if $h(\bar{\theta}) = 0$; and
   b) $\operatorname{rank} \mathrm{D}\, h(\bar{\theta}) = k$.

   The function h is called a local defining function *for $\mathcal{M}$ at θ.*

From now on, we assume $\mathcal{M}$ to be an Euclidean submanifold in the sense of Definition 2.18 unless otherwise stated. We consider the following setting: Let $\mathcal{U}$ be a neighborhood of θ in $\mathcal{E}$ and $\bar{f}$ a real-valued function defined on $\mathcal{U}$. Further, let f be the restriction of $\bar{f}$ to the manifold, that is $f = \bar{f}_{|\mathcal{U} \cap \mathcal{M}}$.

We first investigate the tangent space $T_\theta\mathcal{M}$ for an Euclidean submanifold $\mathcal{M}$. Let γ be a curve in $\mathcal{M}$ with $\gamma(0) = \theta$. Define

$$\gamma'(0) := \lim_{t \to 0} \frac{\gamma(t) - \gamma(0)}{t}.$$

Being a curve in $\mathcal{M}$, γ induces a tangent vector $\dot{\gamma}(0) \in T_\theta\mathcal{M}$. We observe that

$$\dot{\gamma}(0) f = \frac{\mathrm{D}}{\mathrm{d}\, t} f(\gamma(t))|_{t=0} = \frac{\mathrm{D}}{\mathrm{d}\, t} \bar{f}(\gamma(t))|_{t=0} = \mathrm{D}\, \bar{f}(\theta)[\gamma'(0)],$$

which yields the identification of $T_\theta\mathcal{M}$ given by

$$T_\theta\mathcal{M} = \{\gamma'(0)|\gamma : \mathbb{R} \to \mathcal{M} \text{ with } \gamma(0) = \theta\},$$

see [22, Section 3.5]. Thus, $T_\theta\mathcal{M}$ is a linear subspace of $\mathcal{E}$. For an open submanifold $\mathcal{M}$ of $\mathcal{E}$, the tangent space $T_\theta\mathcal{M}$ at a point $\theta \in \mathcal{M}$ is the Euclidean space itself, i.e. $T_\theta\mathcal{M} = \mathcal{E}$. If it is an embedded submanifold in the sense of Definition 2.18, we get that $T_\theta\mathcal{M} = \ker \mathrm{D}\, h(\theta)$, where h is any local defining function for $\mathcal{M}$ at θ. For a proof, we refer to [2, Section 3.5.7].

These considerations yield much easier representations of Riemannian metrics, gradients and Hessians: An Euclidean submanifold can be naturally transferred into a Riemannian manifold by inheriting the standard metric defined on the Euclidean space $\mathcal{E}$: Let $\langle \cdot, \cdot \rangle$ be the Euclidean metric on $\mathcal{E}$. Then, the metric on $\mathcal{M}$ defined at each θ by restriction, i.e. $\langle \xi_\theta, \chi_\theta \rangle_\theta = \langle \xi_\theta, \chi_\theta \rangle$ for $\xi_\theta, \chi_\theta \in T_\theta\mathcal{M}$ is a Riemannian metric [2, Section 3.6.1]. We define the Euclidean submanifold $\mathcal{M}$ endowed with such a metric a *Riemannian submanifold* of $\mathcal{E}$.

In order to compute a gradient of an Euclidean submanifold, we observe that for all $\xi_\theta \in T_\theta\mathcal{M}$, it holds that

$$\langle \xi_\theta, \operatorname{grad} f(\theta) \rangle_\theta = \mathrm{D}\, f(\theta)[\xi_\theta] = \mathrm{D}\, \bar{f}(\theta)[\xi_\theta] = \langle \xi_\theta, \operatorname{grad}^{\mathrm{e}} \bar{f}(\theta) \rangle, \tag{2.7}$$

where $\operatorname{grad}^{\mathrm{e}} \bar{f}(\theta)$ denotes the classical Euclidean gradient of the function $\bar{f}$. We can use this to get an easier presentation of the Riemannian gradient $\operatorname{grad} f(\theta)$, which is an element of $T_\theta\mathcal{M}$: Since $T_\theta\mathcal{M}$ is a subspace of $\mathcal{E}$, there exists a decomposition of $\operatorname{grad} \bar{f}(\theta)$ in $\mathcal{E}$ into a part in $T_\theta\mathcal{M}$ and its orthogonal complement [22, Section 3.8]. Thus, to compute the Riemannian gradient of a Riemannian submanifold of $\mathcal{E}$, we can derive the Euclidean gradient and then orthogonally project it back onto the respective tangent space:

Theorem 2.19. *[2, Section 3.6.1] Let $\mathcal{M}$ be a Riemannian submanifold of $\mathcal{E}$ endowed with the metric $\langle \cdot, \cdot \rangle$. By*

$$\mathrm{Proj}_\theta : \mathcal{E} \to T_\theta\mathcal{M}, \tag{2.8}$$

we denote the projector from $\mathcal{E}$ to $T_\theta\mathcal{M}$, orthogonal with respect to $\langle \cdot, \cdot \rangle$.
For a smooth function $f : \mathcal{M} \to \mathbb{R}$, the Riemannian gradient of f is given by

$$\operatorname{grad} f(\theta) = \mathrm{Proj}_\theta(\operatorname{grad}^{\mathrm{e}} \bar{f}(\theta)),$$

where $\bar{f}$ is a smooth extension of f to a neighborhood of $\mathcal{M}$ in $\mathcal{E}$.

Proof. Since the tangent space $T_\theta\mathcal{M}$ is a subspace of $\mathcal{E}$, we can decompose each tangent vector as the sum of an element in $T_\theta\mathcal{M}$ and its complement which gives the desired expression, see [22, Section 3.8]. □

With this charaterization of the Riemannian gradient, we can derive a similar expression for the Riemannian Hessian: we consider a smooth extension of the Riemannian gradient vector field to $\mathcal{E}$ and project it back onto respective the tangent space.

Theorem 2.20. *[2, Corollary 5.17] Let $\mathcal{M}$ be a Riemannian submanifold of $\mathcal{E}$. Considering a smooth function $f : \mathcal{M} \to \mathbb{R}$, let $\overline{\operatorname{grad} f}$ be a smooth extension of $\operatorname{grad} f$, that is $\overline{\operatorname{grad} f}(\theta)$ is defined in a neighborhood $\mathcal{U}$ of $\operatorname{grad} f(\theta)$ in $\mathcal{E}$ and $\operatorname{grad} f(\theta) = \overline{\operatorname{grad} f}(\theta)_{|\mathcal{U}\cap\mathcal{M}}$. Then, the Riemannian Hessian can be expressed as*

$$\operatorname{Hess} f(\theta)[\xi_\theta] = \operatorname{Proj}_\theta\left(\mathrm{D}\left(\overline{\operatorname{grad} f}(\theta)\right)[\xi_\theta]\right),$$

where $\operatorname{Proj}_\theta$ is the projector from (2.8).

Proof. The Riemannian connection of an Euclidean submanifold is the classical Euclidean gradient vector field projected onto the tangent space, see [22, Section 5.4] for details. The result follows directly from (2.4). □

These characterizations for Euclidean submanifolds help us to derive the necessary concepts for performing Riemannian optimization on the manifold of positive definite matrices in Chapter 4.

In the second part of this chapter, Section 2.2, we introduce the concepts of Riemannian optimization based on the differential-geometric foundations introduced in this section.

2.2 Riemannian Optimization Methods

In the previous section, we have built the main basis for solving optimization problems of the form

$$\min_{\theta\in\mathcal{M}} f(\theta), \tag{2.9}$$

where $\mathcal{M}$ is a smooth Riemannian manifold and $f : \mathcal{M} \to \mathbb{R}$ is a smooth function. In this section, we will first introduce analogies to Euclidean optimization theory for necessary and sufficient conditions for local optima of f. In Section 2.2.2, we introduce tools that allow to move an manifolds which helps us to build optimization algorithms in the subsequent sections, Sections 2.2.3 - 2.2.5. We will discuss the Riemannian trust-region algorithm in detail in Section 2.2.4 as it is central to Chapters 5 and 6.

2.2.1 Optimality on Manifolds and Convergence of Algorithms

We first define *local* and *global optima* for problem (2.9):

Definition 2.21. (global and local minimizer, global and local minimum)*[22, Section 4.2] We consider the optimization problem* (2.9)*.*

1. *A point $\theta^* \in \mathcal{M}$ is called a* global minimizer *to problem* (2.9)*, if*
$$f(\theta^*) \leq f(\theta) \quad \forall \theta \in \mathcal{M}.$$
The value $f(\theta^*)$ is called a global minimum *of f in $\mathcal{M}$. If $f(\theta^*) < f(\theta)$ for all $\theta \in \mathcal{M} \backslash \{\theta^*\}$, we say that θ^* is a* strict global minimizer *and $f(\theta^*)$ is a* strict global minimum *of f in $\mathcal{M}$.*

2. *A point θ^* is called a* local minimizer *to problem* (2.9)*, if there exists a neighborhood $\mathcal{U}$ of θ in $\mathcal{M}$ such that*
$$f(\theta^*) \leq f(\theta) \quad \forall \theta \in \mathcal{U}.$$
The value $f(\theta^*)$ is called a local minimum *of f in $\mathcal{M}$. If θ^* is a local minimizer of f in $\mathcal{M}$ and $f(\theta^*) < f(\theta)$ for $\theta \in \mathcal{U} \backslash \{\theta^*\}$, then θ^* is called a* strict local minimizer*.*

A few remarks are in order:

(i) Throughout this section, we assume that f is lower-bounded, such that the expression (2.9) is well-defined.

(ii) Similar to the Euclidean case, our framework allows to work with maximization problems of the form
$$\max_{\theta \in \mathcal{M}} f(\theta), \tag{2.10}$$
for $f : \mathcal{M} \to \mathbb{R}$ smooth.

(iii) In the case of a maximization problem of the form (2.10), we define (strict) *global* and *local maximizers* as in Definition 2.21 by exchanging the " $\leq$ "," $<$ " signs by " $\geq$ "," $>$ " signs. A maximization problem (2.10) can be transformed into a minimization problem (2.9), since
$$\arg\max_{\theta \in \mathcal{M}} f(\theta) = -\arg\min_{\theta \in \mathcal{M}} (-f(\theta))$$
holds true.

In analogy to the Euclidean gradient, the Riemannian gradient plays a central role for optimization theory. The first important fact is that local minimizers are *critical points*, that is the gradient vanishes in that point (first-order necessary condition):

Theorem 2.22. *[120, Theorem 3.4] Let $f : \mathcal{M} \to \mathbb{R}$ be a smooth function and $\theta^* \in \mathcal{M}$ a local minimizer of f. Then the gradient at θ^* vanishes, that is*

$$\operatorname{grad} f(\theta^*) = 0.$$

Proof. We refer to [120, Section 3.3]. □

Theorem 2.22 is the basis for the optimization algorithms presented in this thesis: We seek to build iterates converging to a point θ^* at which the gradient vanishes. Another important fact of the gradient is used to build optimization methods: The Riemannian gradient is the steepest-ascent direction of f at θ, making it important for various optimization methods:

$$\arg \max_{\substack{\xi_\theta \in T_\theta \mathcal{M} \\ \|\xi_\theta\|_\theta = 1}} \mathrm{D} f(\theta)[\xi] = \frac{\operatorname{grad} f(\theta)}{\|\operatorname{grad} f(\theta)\|_\theta},$$

where $\|\cdot\|_\theta = \sqrt{\langle \cdot, \cdot \rangle_\theta}$ [2, Section 3.6].

In accordance with Euclidean optimization theory, the Riemannian Hessian characterizes the *second-order necessary* and *sufficient optimality condition* for the Riemannian setting:

Theorem 2.23. *[22, Proposition 6.2 & 6.3] Consider the optimization problem* (2.9)*. The following statements hold true.*

i) If θ^* is a local minimizer of f, then the Hessian at θ^* is positive semidefinite, i.e. $\operatorname{Hess} f(\theta^*)[\xi_{\theta*}] \geq 0$ *for all $\xi_{\theta^*} \in T_{\theta^*}\mathcal{M}$ (second-order necessary condition).*

ii) If $\operatorname{grad} f(\theta^*) = 0$ *and* $\operatorname{Hess} f(\theta^*)[\xi_{\theta^*}] > 0$ *for all $\xi_{\theta^*} \in T_{\theta^*}\mathcal{M}$, $\theta^* \in \mathcal{M}$, then θ^* is a local minimizer of f (second-order sufficient condition).*

Proof. For a proof, see [120, Section 3.3]. □

Geodesic convexity

Most algorithms are designed to converge to local minima, in general, we cannot know whether there is a better minimum in the domain and heuristics are needed to find a global minimum. In the Euclidean setting however, we can approach global minima if we have a convex optimization problem, meaning that both the set of constraints and the function is convex [105, Chapter 1]. An analogue can be derived for the Riemannian setting by *geodesic convexity*: we characterize convex sets by special geodesics joining two points in a set being fully contained in the set. For convex functions defined on a manifold, we consider the concatenation of the function defined on a manifold with the geodesic. The subsequent definition formalizes this generalization:

Definition 2.24. (geodesically convex sets, geodesically convex functions) *[22, Definition 11.2 & 11.3] Let N be a subset of a manifold $\mathcal{M}$ and $f : N \to \mathbb{R}$ a smooth function.*

1. *The subset N is called* geodesically convex *if for every $\theta_1, \theta_2 \in N$, there exists a geodesic $\gamma : [0,1] \to \mathcal{M}$ such that $\gamma(0) = \theta_1$, $\gamma(1) = \theta_2$ and $\gamma(t)$ is in N for all $t \in [0,1]$.*
2. *The function f is called* geodesically convex *if the set N is geodesically convex and $f \circ \gamma : [0,1] \to \mathbb{R}$ is Euclidean convex for any geodesic segment $\gamma : [0,1] \to \mathcal{M}$ whose image is in N, i.e. for all $\theta_1, \theta_2 \in N$ and all geodesics γ connecting θ_1 and θ_2 in N,*
$$f(\gamma(t)) \leq (1-t)f(\theta_1) + tf(\theta_2)$$
for all $t \in [0,1]$.
3. *We say $f : S \to \mathbb{R}$ is* geodesically concave *if $-f$ is geodesically convex.*
4. *If f is both geodesically convex and geodesically concave, it is called* geodesically linear.

Geodesic convexity of an objective function f ensures that any local minimizer is a global maximizer, which is an analogy to Euclidean spaces:

Theorem 2.25. *[22, Theorem 11.6] If $f : \mathcal{M} \to \mathbb{R}$ is geodesically convex, then any local minimizer is a global minimizer.*

Proof. We refer to [22, Section 11.2] for a proof. □

Convergence on manifolds

With the characterization of local and global minima, we seek to build algorithms that produce iterates converging to a minimum of an objective. Formally, an infinite sequence

$\{\theta^t\}_{t\in\mathbb{N}_0}$ is said to be *convergent* if there exists a chart $(\mathcal{U},\varphi)$ of $\mathcal{M}$, a point $\theta^* \in \mathcal{U}$ and an integer $T > 0$ such that θ^t is in $\mathcal{U}$ for all $t \geq T$ and such that the sequence $\{\varphi(\theta^t)\}_{t=T,T+1,\dots}$ converges to $\varphi(\theta^*)$ [2, Section 4.5.1]. The point $\varphi^{-1}(\lim_{t\to\infty}(\theta^t)) \in \mathcal{M}$ is called the *limit* of the convergent sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$. Due to the Hausdorff property of a manifold, any convergent sequence of a manifold has one and only one limit point, see Appendix A.1. Further, given a sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$, we say that θ^* is an *accumulation point* or *limit point* if there exists a subsequence $\{\theta^{j_t}\}_{t\in\mathbb{N}_0}$ that converges to θ^*.
Algorithms that produce convergent sequences with limit points being local minima are of major interest and algorithms achieving this fast are desirable, for which reason we introduce the *order of convergence* in analogy to optimization theory in Euclidean spaces:

Definition 2.26. (linear, superlinear convergence, convergence of order p) *[2, Section 4.5.1] Let $\mathcal{M}$ be a Riemannian manifold and let* dist *denote the Riemannian distance on $\mathcal{M}$. Let $\{\theta^t\}_{t\in\mathbb{N}_0}$ be a sequence converging to a limit point $\theta^* \in \mathcal{M}$.*

1. *We say that the sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$ converges* linearly *to a point $\theta^* \in \mathcal{M}$ if there exists a constant $c \in (0,1)$ and an integer $T \geq 0$ such that, for all $t \geq T$, it holds that*
$$\operatorname{dist}(\theta^{t+1},\theta^*) \leq c \operatorname{dist}(\theta^t,\theta^*) \tag{2.11}$$

2. *An iterative algorithm on $\mathcal{M}$ is said to* converge locally linearly *to a point θ^* if there exists a neighborhood $\mathcal{V}$ of θ^* and a constant $c \in (0,1)$ such that for every initial point $\theta^0 \in \mathcal{V}$, the sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$ generated by the algorithm satisfies* (2.11). *The constant c is called the* convergence rate.

3. *Let $(\mathcal{U},\varphi)$ be a chart of $\mathcal{M}$ with $\theta \in \mathcal{U}$. If*
$$\lim_{k\to\infty} \frac{\|\varphi(\theta^{t+1}) - \varphi(\theta^*)\|}{\|\varphi(\theta^t) - \varphi(\theta^*)\|} = 0,$$
then $\{\theta^t\}_{t\in\mathbb{N}_0}$ is said to converge superlinearly *to θ^*. If there exist constants $p > 0$, $c \geq 0$ and $T \geq 0$ such that for all $t \geq T$ it holds that*
$$\|\varphi(\theta^{t+1}) - \varphi(\theta^*)\| \geq c\|\varphi(\theta^t) - \varphi(\theta^*)\|^p, \tag{2.12}$$
then $\{\theta^t\}_{t\in\mathbb{N}_0}$ is said to converge to θ^* with order *at least p.*

4. *An iterative algorithm on $\mathcal{M}$ is said to* converge locally *to a point θ^** with order at least p *if there exists a chart $(\mathcal{V},\varphi)$ at θ^* and a constant $c > 0$ such that, for every initial point $\theta^0 \in \mathcal{V}$, the sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$ generated by the algorithm satisfies* (2.12). *If $p = 2$, the convergence rate is said to be* quadratic.

2.2.2 Moving on Manifolds and Line-search Methods

Equipped with sufficient and necessary conditions for local minima on manifolds and a notion of convergence speed, we are in the position to construct optimization algorithms, that is, to generalize numerical algorithms designed for unconstrained Euclidean optimization to optimization problems on manifolds of the form (2.9). Typical algorithms in unconstrained nonlinear optimization in $\mathbb{R}^d$ consist of *line-search methods*, where iterates θ^{t+1} at iteration $t+1$ are updated by the formula

$$\theta^{t+1} = \theta^t + \beta^t v^t, \tag{2.13}$$

where $v^t \in \mathbb{R}^d$ is the *search direction* and $\beta^t \in \mathbb{R}$ is the *step size* [105, Chapter 3]. The step size and the search direction are typically chosen such that the iterates converge to a critical point. For example, the famous gradient descent algorithm chooses the search direction v^t as the negative gradient at point θ^t [105, Section 3.3]. If we want to generalize the gradient descent method to the manifold setting, v^t could be chosen as $-\operatorname{grad} f(\theta^t)$, being a tangent vector to θ^t. However, the expression "$\theta^t + \beta^t v^t$" might not be well-defined on a manifold. Thus, we need a tool on how to move on manifolds such that within a Riemannian version of line-search methods like gradient descent, we produce new iterates that are still elements of the respective manifolds [22, Section 3.6]. The key idea to construct such algorithms consists of performing the search along a curve in $\mathcal{M}$ whose tangent vector at $t = 0$ is equal to v^t. The choice of such a curve is where the *exponential map* and *retractions* come into play.

We first consider the special curves on $\mathcal{M}$ that are geodesics. One can show that for every tangent vector $\xi_\theta \in T_\theta\mathcal{M}$, there exists an interval I around 0 and a unique geodesic $\gamma^{gd} \in \Gamma$, $\gamma^{gd}(t;\theta,\xi_\theta) : I \to \mathcal{M}$, such that $\gamma(0) = \theta$ and $\dot{\gamma}(0) = \xi_\theta$, [80, Theorem 4.10]. Moreover, the homogenity property $\gamma^{gd}(t;\theta,a\xi_\theta) = \gamma^{gd}(at;\theta,\xi_\theta)$ holds [2, Section 5.4]. This geodesic characterizes the *exponential map*:

Definition 2.27. (exponential map) *[2, Section 5.4] The mapping*

$$\operatorname{Exp}_\theta : T_\theta\mathcal{M} \to \mathcal{M} : \xi_\theta \mapsto \operatorname{Exp}_\theta(\xi_\theta) = \gamma^{gd}(1;\theta,\xi_\theta)$$

is called the exponential map *at θ. In particular,* $\operatorname{Exp}_\theta(0) = \theta$.

With the exponential map, we are equipped with a natural tool to construct new iterates on a manifold via a generalization of the line-search method: By applying the exponential map on $\beta^t v^t$, where $v^t \in T_{\theta^t}\mathcal{M}$, we move along a curve with a velocity

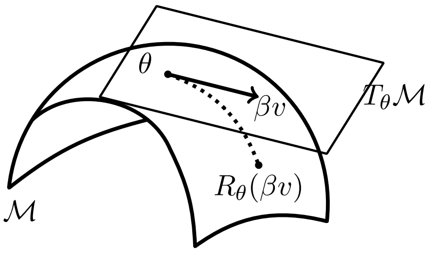


Figure 2.1: Retraction R on a manifold $\mathcal{M}$

equal to the search direction. At the same time, if we do not make any step, that is $\beta^t = 0$, we remain in the point θ^t, this is the so-called *rigidity condition* [2, Section 4.1]. These are favorable properties of the exponential map since they generalize the Euclidean line-search method in both intuitions. Though, in many cases, the exponential map is very expensive to compute resulting in slow Riemannian optimization methods. Recent literature on Riemannian optimization [2, 22, 120] is thus based on *retractions*: these are curves equipped with the same favorable properties as exponential maps.

Definition 2.28. (retraction)*[2, Definition 4.1.1.] A* retraction *on a manifold $\mathcal{M}$ is a smooth map $R : T\mathcal{M} \to \mathcal{M}$ with the following properties. For each $\theta \in \mathcal{M}$, let $R_\theta : T_\theta\mathcal{M} \to \mathcal{M}$ be the restriction of R at θ. Then,*

1. *$R_\theta(0) = \theta$ (rigidity condition), and*
2. *$\mathrm{D}\, R_\theta(0) : T_\theta\mathcal{M} \to T_\theta\mathcal{M}$ is the identity map: $\mathrm{D}\, R_\theta(0)[\xi_\theta] = \xi_\theta$.*

Note that the exponential map is a special choice of a retraction and thus for every Riemannian manifold, there exists at least one retraction. With the definition of a retraction, we can thus generalize the Euclidean line-search update $\theta^{t+1} = \theta^t + \beta^t v^t$ by using $\theta^{t+1} = R_{\theta^t}(\beta^t v^t)$ to produce a new update on a manifold, see Figure 2.1.

Besides its ability to transform points from $T_\theta\mathcal{M}$ to $\mathcal{M}$, retractions have an additional purpose: they allow to transform cost functions defined in a neighborhood of $\theta \in \mathcal{M}$ into cost functions defined on the vector space $T_\theta\mathcal{M}$ [22, Section 3.6]. To be more precise, we introduce the curve $\gamma : \mathbb{R} \to \mathcal{M} : t \mapsto R_\theta(tv)$ for a retraction R and a tangent vector $v \in T_\theta\mathcal{M}$. Let g be a smooth function defined by $g : \mathbb{R} \to \mathbb{R} : t \mapsto f(\gamma(t))$. By a first-order Taylor expansion of g around 0, we get

$$g(t) = f(\gamma(t)) = f(\theta) + t\langle \operatorname{grad} f(\theta), v\rangle_\theta + \mathcal{O}(t^2) \tag{2.14}$$

due to the rigidity condition of the retraction. By setting $\xi = tv$, we introduce $\hat{f} = f \circ R$, denoted the *pullback* of the tangent spaces via f and R. It is a smooth function from $T\mathcal{M}$ to $\mathbb{R}$, and

$$\hat{f}_\theta = f \circ R_\theta$$

denotes the restriction of $\hat{f}$ to $T_\theta\mathcal{M}$. With (2.14), we get

$$\hat{f}_\theta(\xi) = f(R_\theta(\xi)) + \langle \operatorname{grad} f(\theta), \xi \rangle_\theta + \mathcal{O}(\|\xi\|_\theta^2). \tag{2.15}$$

Algorithm 1: Line-search methods on manifolds [2, Section 4.2]

```
   Input: objective $f$, initial iterate $\theta^0 \in \mathcal{M}$, retraction $R$
   Output: sequence of parameters $\{\theta^t\}$
 1 for $t = 0, 1, 2, \ldots$ do
 2     Choose appropriate $\beta^t$, $v^t \in T_{\theta^t}\mathcal{M}$ such that there is a decay in the objective
 3     Set
           $\theta^{t+1} = R_{\theta^t}(\beta^t v^t)$
 4 end for
```

The expression (2.14) is central to Riemannian line-search methods, as it allows to generalize convergence theory of Euclidean line-search methods onto the manifold setting. It can be shown that under suitable conditions on the step length β^t in Algorithm 1, the algorithm converges to a critical point with linear speed. A precise characterization is beyond the scope of this thesis, we refer the reader to [2, Sections 4.3-4.5] and [22, Chapter 4] for a detailed investigation.

2.2.3 Second Order Taylor's Expansion on Curves and Newton's Algorithm

So far we have considered algorithms based on the gradient, only. In the Euclidean setting, optimization algorithms benefit from second-order information via the Hessian and yield faster local convergence rates [105]. We thus derive a generalization of Newton's method for the Riemannian case.

We again consider the concatenation $g : \mathbb{R} \to \mathbb{R} : t \mapsto f(\gamma(t))$, where γ denotes the curve associated with an arbitrary retraction R. With (2.14), the second-order Taylor expansion of g around 0 reads

$$g(t) = f(\theta) + t\langle \operatorname{grad} f(\theta), v\rangle_\theta + \frac{t^2}{2}\ddot{\gamma}(0) + \mathcal{O}(t^3). \tag{2.16}$$

For the second last term in (2.16), we get

$$\begin{aligned}\ddot{\gamma}(0) &= \frac{\mathrm{D}}{\mathrm{d}\,t}\langle \operatorname{grad} f(\gamma(t)), \dot{\gamma}(t)\rangle_{\gamma(t)} \\ &= \langle \operatorname{Hess} f(\gamma(t))[\dot{\gamma}(t)], \dot{\gamma}(t)\rangle_{\gamma(t)} + \langle \operatorname{grad} f(\gamma(t)), \ddot{\gamma}(t)\rangle_{\dot{\gamma}(t)}\end{aligned}$$

by the Leibniz property of the Riemannian connection (Definition 2.9) and hence

$$\begin{aligned}g(t) = f(\gamma(t)) = f(\theta) + t\langle \operatorname{grad} f(\theta), v\rangle_\theta &+ \frac{t^2}{2}\langle \operatorname{Hess} f(\theta)[v], v\rangle_\theta \\ &+ \frac{t^2}{2}\langle \operatorname{grad} f(\theta), \ddot{\gamma}(0)\rangle_\theta + \mathcal{O}(t^3)\end{aligned} \tag{2.17}$$

by the ridigity property of the retraction [22, Section 5.9]. The expression (2.17) does not fully generalize the Euclidean second-order taylor expansion due to the term

$$\frac{t^2}{2}\langle \operatorname{grad} f(\theta), \ddot{\gamma}(0)\rangle_\theta.$$

It is of order t^2 and depends on γ (and thus on the retraction) which is unfortunate [22, Section 6.2]. However, this term vanishes if θ is a critical point or if $\ddot{\gamma}(0)$ equals zero. This leads us to the following definition of a *second-order retraction*:

Definition 2.29. (second-order retraction) *[2, Section 5.5] A* second-order retraction *R on a Riemannian manifold $\mathcal{M}$ is a retraction such that for all $\theta \in \mathcal{M}$ and $\xi_\theta \in T_\theta\mathcal{M}$, the curve $\gamma(t) = R_\theta(tv)$ has zero acceleration at $t = 0$, i.e. $\ddot{\gamma}(t) = 0$.*

Note that the exponential map is a second-order retraction as the according curve γ is a geodesic.

With a second-order retraction R, the Taylor expansion (2.17) provides a quadratic approximation $m_\theta : T_\theta\mathcal{M} \to \mathbb{R}$ for the function $f \circ R_\theta$ given by

$$m_\theta(\xi) = f(\theta) + \langle \operatorname{grad} f(\theta), \xi\rangle_\theta + \frac{1}{2}\langle \operatorname{Hess} f(\theta)[\xi], \xi\rangle_\theta. \tag{2.18}$$

Thus, if the Riemannian Hessian is positive definite, we can minimize m_θ to search for a minimum of f. Since m_θ is smooth, any minimizer must be a critical point of m_θ. We get

$$\operatorname{grad} m_\theta(\xi) = \operatorname{grad} f(\theta) + \operatorname{Hess} f(\theta)[\xi],$$

which vanishes if and only if the *Riemannian Newton equation*

$$\operatorname{Hess} f(\theta)[\xi] = -\operatorname{grad} f(\theta) \tag{2.19}$$

is fulfilled [2, Section 6.2].

This yields the Riemannian Newton algorithm, see Algorithm 2.

Algorithm 2: Riemannian Newton method [2, p. 113]

```
  Input: objective $f$ with Riemannian gradient $\operatorname{grad} f$ and Hessian $\operatorname{Hess} f$ , initial
         iterate $\theta^0 \in \mathcal{M}$, retraction $R$
  Output: sequence of parameters $\{\theta^t\}$
1 for $t = 0, 1, 2, \dots$ do
2     Solve the Newton equation
          $\operatorname{Hess} f(\theta^t)[\xi^t] = -\operatorname{grad} f(\theta^t).$
      for the unknown $\xi^t \in T_{\theta^t}\mathcal{M}$.
3     Set
          $\theta^{t+1} = R_{\theta^t}(\xi^t)$
4 end for
```

It can be shown that under suitable conditions, the Riemannian Newton algorithm converges locally superlinearly to a critical point:

Theorem 2.30. *[2, Theorem 6.3.2] Under the requirements and notation of Algorithm 2, assume that there exists $\theta^* \in \mathcal{M}$ such that* $\operatorname{grad} f(\theta^*) = 0$ *and* $\operatorname{Hess} f(\theta^*)$ *is invertible. Then there exists a neighbourhood $\mathcal{U}$ of θ^* in $\mathcal{M}$ such that for all $\theta^0 \in \mathcal{U}$, Algorithm 2 generates an infinite sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$ converging superlinearly (at least quadratically) to θ^*.*

Proof. We refer to [2, Section 6.3] for a proof. □

Note that this result holds for general retractions, in particular for retractions that are not second-order. This is due to the fact that

$$\operatorname{Hess} f(\theta^*) = \operatorname{Hess}(f \circ R_{\theta^*})(0)$$

holds true for any critical point θ^* of f [2, Section 6.3].

2.2.4 The Riemannian Trust-Region Method

We have introduced a Riemannian version of the Newton algorithm which gives us local superlinear convergence. However, the shortcomings of the Euclidean Newton method transfer to the Riemannian setting [2, Chapter 7]. One of the major drawbacks is that Newton's method is not globally convergent, meaning that convergence to a critical point from any starting point is not ensured. Convergence is ensured only in a neighborhood of a critical point but this neighborhood may be arbitrarily small. One key wish thus consists in globalizing Newton's method such that it converges to a critical point independently from the starting point. Furthermore, Theorem 2.30 ensures local convergence to critical points, only. From an optimization perspective, however, only the cases where $\operatorname{Hess} f(\theta^*) \succ 0$ are beneficial since we wish to avoid converging to local maxima or saddle points [2, Chapter 7].

A remedy to the drawbacks of the Riemannian Newton method consists in considering *Riemannian trust-region methods* which are globally convergent methods to local minima while at the same time they preserve the local convergence properties of Newton's method under suitable conditions [1]. They are based on a generalization of the quadratic model (2.18), that is we consider the quadratic model

$$f(R_\theta(s)) \approx m_\theta(s) = f(\theta) + \langle \operatorname{grad} f(\theta), s\rangle_\theta + \frac{1}{2}\langle H_\theta[s], \xi\rangle_\theta, \tag{2.20}$$

where $s \in T_\theta\mathcal{M}$ and H_θ is allowed to be any linear operator on $T_\theta\mathcal{M}$ that is self-adjoint, that is $\langle H_\theta[s], v\rangle_\theta = \langle H_\theta[v], s\rangle_\theta$ for all s, v in $T_\theta\mathcal{M}$ [1]. A typical, suitable choice for H_θ is the Riemannian Hessian $\operatorname{Hess} f(\theta)$, which then results in the second-order Taylor expansion given by (2.17). If the Riemannian Hessian is used in (2.20), we call the method presented in the following a *Riemannian Newton trust-region method (R-NTR).*

The main idea of the Riemannian trust-region method is not to compute the critical points of m_θ directly, but instead minimizing the quadratic model m_θ in every step. Since the model is only a local approximation of the pullback $\hat{f}$ around θ, we further restrict the minimization problem to a ball around the origin in the tangent space, to the so-called *trust-region* [1, 105]. At iteration t, we solve the problem

$$\min_{s \in T_{\theta^t}\mathcal{M}} m_{\theta^t}(s) \quad \text{subject to } \|s\|_{\theta^t} \leq \Delta_t. \tag{2.21}$$

for some $\Delta_t > 0$ (*trust-region radius*) to obtain a descent direction at iteration t. For better readability, we write m_t for m_{θ^t} in the following. The quadratic problem in (2.21) is called the *trust-region subproblem* and is a constrained quadratic optimization problem defined on the tangent space $T_{\theta^t}\mathcal{M}$ with the inner product $\langle \cdot, \cdot \rangle_{\theta^t}$. The solution s to the subproblem (2.21) is then mapped back onto the manifold $\mathcal{M}$ via a retraction, that is the next tentative iterate θ^+ is given by $\theta^+ = R_{\theta^t}(s)$. Depending on how well the model m_t approximates the local pullback $\hat{f}$ in $T_{\theta^t}\mathcal{M}$ in the neighborhood of $0_{\theta^t} \in T_{\theta^t}\mathcal{M}$, the tentative iterate is accepted ($\theta^{t+1} = \theta^+$) or rejected ($\theta^{t+1} = \theta^t$). To quantify the model quality at iterate t, the ratio of actual to model improvement is computed [2, Section 7.2.2]:

$$\rho_t = \frac{f(\theta^t) - f(\theta^+)}{m_t(0) - m_t(s)} = \frac{f(\theta^t) - f(R_{\theta^t}(s))}{m_t(0) - m_t(s)}. \tag{2.22}$$

If ρ_t is below a pre-specified threshold ρ', the local model is deemed inaccurate and the step must be rejected. In order to get a globally convergent algorithm, this factor ρ' must be no larger than 1/4, see Theorem 2.31. Furthermore, the trust-region radius Δ_t must be reduced by a factor $\tau_1 < 1$. Conversely, if ρ_t is close to 1, the model m_t is a very good fit for the pullback $\hat{f}$ and the trust-region radius can be increased in order to be able to perform bigger steps in the next iteration. In between these two extremes, if ρ_t is small but above the threshold ρ', the step typically is accepted but the trust-region is scaled down [1]. In case $\rho \gg 1$, the overall optimization is producing a big decrease despite a poor model fit. Then, we accept the step and still increase the trust-region radius as long as we stay below a a predefined bound $\bar{\Delta} > 0$ [22, Section 6.4]. The overall procedure is summarized in Algorithm 3.

Typical choices of the parameters are $\omega_1 = 0.25$, $\omega_2 = 0.75$ and $\tau_1 = 0.25$, $\tau_2 = 2$, a detailed discussion of these parameters for the Euclidean trust-region method can be found in [34, Chapter 17].

One can show that the Riemannian trust-region algorithm converges to a critical point independently of the starting point if the rejection threshold ρ' is below 1/4 and additional regularity assumptions are met, see [2, Theorem 7.4.4]. These assumptions are in particular fulfilled in case the *level set* is compact.

Theorem 2.31. *[2, Section 7.4.1] Let $\mathcal{M}$ be a manifold and $f : \mathcal{M} \to \mathbb{R}$ be smooth and bounded from below. Consider Algorithm 3 with starting point θ^0 and $\rho' < 1/4$, and let $\{\theta^t\}_{t\in\mathbb{N}_0}$ be a sequence of iterates generated by the algorithm. If the sequence $\{\theta^t\}_{t\in\mathbb{N}_0}$ produced by Algorithm 3, where the quadratic subproblem is solved by tCG (Algorithm 4),* *stays in a compact set, it holds that*

$$\lim_{t\to\infty} \operatorname{grad} f(\theta^t) = 0.$$

Proof. The proof is a generalization of the proof for global convergence of the trust-region method in the Euclidean setting [105, Section 4.3] and we refer to [2, Section 7.4.1] for the proof in the manifold setting. □

Algorithm 3: Riemannian trust-region method [2, p. 142]

```
   Input: objective $f$ with linear operator $H$, retraction $R$, initial iterate $\theta^0 \in \mathcal{M}$,
          initial TR-radius $\Delta_0$, maximal TR-radius $\bar{\Delta}$, rejection threshold
          $\rho' \in [0, 1/4)$, acceptance parameters $0 \le \omega_1 \le \omega_2 \le 1, \tau_1 < 1, \tau_2 > 1$
   Output: sequence of parameters $\{\theta^t\}$
 1 for $t = 0, 1, 2, \dots$ do
 2     Obtain $s^t$ by (approximately) solving the TR-subproblem
           $\min_{s \in T_{\theta^t}\mathcal{M}} \hat{m}_{\theta^t}(s) = f(\theta^t) + \langle \operatorname{grad} f(\theta^t), s\rangle_{\theta^t} + \frac{1}{2}\langle H_t[s], s\rangle_{\theta^t}$   s.t. $\langle s, s\rangle_{\theta^t} \le \Delta_t^2$;
 3     Evaluate $\rho_t = \frac{f(\theta^t) - f(R_{\theta^t}(s^t))}{\hat{m}_{\theta^t}(0_{\theta^t}) - \hat{m}_{\theta^t}(s^t)}$
 4     if $\rho_t < \omega_1$ then
 5         $\Delta_{t+1} = \tau_1 \Delta_t$;
 6     else if $\rho_t > \omega_2$ and $\|s^t\|_{\theta^t} = \Delta_t$ then
 7         $\Delta_{t+1} = \min(\tau_2 \Delta_t, \bar{\Delta})$;
 8     else
 9         $\Delta_{t+1} = \Delta_t$;
10     if $\rho_t > \rho'$ then
11         $\theta^{t+1} = R_{\theta^t}(s^t)$;
12     else
13         $\theta^{t+1} = \theta^t$
14     set $t = t + 1$;
15 end for
```

The crucial part in the Riemannian trust-region method consists in solving the quadratic subproblem (2.21) efficiently in every step. Since this is a problem posed on a tangent space and thus a vector space endowed with the inner product $\langle \cdot, \cdot \rangle_\theta$, classical algorithms to solve constrained quadratic problems can be applied. Methods to solve this can be mainly divided into exact methods and methods based on finding an approximate solution, an overview can be found in [34, Chapter 7]. We call the iterations of algorithms to solve the subproblem *inner iterations*, whereas the iterations of the overall trust-region algorithm (Algorithm 3) are called *outer iterations*.

We here present an efficient way to approximately solve the quadratic problem, the *truncated conjugate gradient method* (tCG) from Steihaug [132]. This method is especially favorable for the Riemannian setting proposed here, as the dimension of the tangent space usually is large, especially in case of matrix manifolds. Besides its advantages for large-scale problems, the truncated conjugate gradient method is *matrix-free*, meaning that it is suited for problems where the Hessian is given as a linear operator, only [34, Section 7.5.1]. This makes it a favorable method to solve the quadratic subproblem in Riemannian trust-region methods. The key idea of truncated conjugate gradient is that if we stay within the trust-region radius, the subproblem (2.21) coincides with the second-order Taylor expansion (2.17), where $\operatorname{Hess} f(\theta^t)$ is replaced by H_t. If H_t is positive definite, we can thus try to solve the Newton equations (2.19) until we leave the trust-region. If we are within the trust-region, the process of building new iterates is borrowed from the well-known *conjugate gradient method*, an iterative method to solve large linear systems of equations with a symmetric, positive definite operator. A detailed discussion of the CG method can be found in [105, Chapter 5]. In case we face negative curvature at an iterate, we can no longer assume that H_t is positive definite and thus we need to stop with generating iterates by the conjugate gradient method. Similarly, if a step is larger than the trust-region radius, we do not trust the quadratic model anymore. In both cases, we compute the next inner iteration by following the potential next conjugate direction until we reach the boundary of the trust-region radius, that is we *truncate* the conjugate direction at the boundary [22, Section 6.5].
Optionally, a preconditioner M_t can be used in order to achieve a faster convergence behavior of the method (see [34, Section 5.1.6] for details). The preconditioned tCG method is summarized in Algorithm 4.

It can be shown that for iterates $s_n, n = 0, 1, 2, \ldots$, the quadratic model m_t decreases [2, Proposition 7.3.2]. However, note that the quadratic subproblem must be solved in every outer iteration, so early stopping of the truncated conjugate gradient method is desirable [22, Section 6.5]. The simplest stopping criterion to use is to stop after a fixed number of iterations, whereas the stopping criterion

$$\|z_{n+1}\| \leq \|z_0\| \min(\|z_0\|^\delta, \kappa) \tag{2.23}$$

Algorithm 4: Truncated conjugate gradient method on a tangent space [2, p. 144]

```
Input: Riemannian gradient $\operatorname{grad} f(\theta^t)$, linear operator $H_t$, inner product
       $\langle \cdot, \cdot \rangle_{\theta^t}$, optional preconditioner $M_t$, TR radius $\Delta_t$, termination
       parameter $\delta$, $\kappa$
Output: sequence of parameters $\{s_n\}$
1  Set $s_0 = 0$, $r_0 = \operatorname{grad} f(\theta^t)$, $z_0 = M_t r_0$, $p_0 = -z_0$;
2  for $n = 0, 1, 2, \dots$ do
3      if $\langle p_n, H_t[p_n] \rangle_{\theta^t} \leq 0$ then
4          Compute $\tau^{tCG} = \arg\min_{\tau} \hat{m}_{\theta^t}(s_n + \tau p_n)$ s.t. $\|s_n + \tau p_n\|_{\theta^t} = \Delta_t$;
5          Set $s_{n+1} = s_n + \tau^{tCG} p_n$;
6          return $s_{n+1}$;
7      Compute $\alpha_n^{tCG} = \frac{\langle r_n, z_n \rangle_{\theta^t}}{\langle p_n, H_t[p_n] \rangle_{\theta_t}}$;
8      if $\|s_n + \alpha_n^{tCG} p_n\|_{\theta^t} > \Delta_t$ then
9          Compute $\tau^{tCG} = \arg\min_{\tau} \hat{m}_{\theta^t}(s_n + \tau p_n)$ s.t. $\|s_n + \tau p_n\|_{\theta^t} = \Delta_t$;
10         Set $s_{n+1} = s_n + \tau^{tCG} p_n$;
11         return $s_{n+1}$;
12     else
13         $s_{n+1} = s_n + \alpha_n^{tCG} p_n$;
14         $r_{n+1} = r_n + \alpha_n^{tCG} H_t[p_n]$;
15         $z_{n+1} = M_t r_{n+1}$;
16         $\beta_{n+1}^{tCG} = \frac{\langle z_{n+1} z_{n+1} \rangle_{\theta^t}}{\langle z_n, z_n \rangle_{\theta^t}}$;
17         $p_{n+1} = -z_{n+1} + \beta_{n+1}^{tCG} p_n$;
18         if $\|z_{n+1}\|_{\theta^t} \leq \|z_0\|_{\theta^t} \min(\|z_0\|_{\theta^t}^{\delta}, \kappa)$ then
19             return $s_{n+1}$;
20 end for
```

for $\delta > 0$, $\kappa \geq 0$ real parameters chosen in advance, ensures superlinear convergence of the Riemannian trust-region problem, see Theorem 2.32.

The following result gives details about the local convergence rate of the Riemannian trust-region algorithm (Algorithm 3), where the subproblem is solved by the truncated conjugate gradient method (Algorithm 4):

Theorem 2.32. *[2, Theorem 7.4.11 & Theorem 7.4.12] Consider Algorithm 3, where the subproblem is solved by Algorithm 4 with stopping criterion* (2.23)*. Suppose that f is smooth, R is a second-order retraction and that*

$$\|H_t - \mathrm{Hess}(f \circ R_{\theta^t})(0_{\theta^t})\| \leq \beta_H \|\mathrm{grad}\, f(\theta^t)\|$$

for some constant β_H. Let $\theta^* \in \mathcal{M}$ be a local minimizer of f with positive definite Hessian $\mathrm{Hess}\, f(\theta^*)$*. Further, assume that* $\mathrm{Hess}(f \circ R_{\theta^t})(0_{\theta^t})$ *is Lipschitz continous at 0_{θ^t} uniformly in θ in a neighborhood of θ^*, i.e. there exists $\beta_{L2} > 0, \epsilon_1 > 0$ and $\epsilon_2 > 0$, such that for all $\theta \in B_{\epsilon_1}(\theta^*)$ and all $\xi \in B_{\epsilon_2}(0_\theta)$ it holds that*

$$\|\mathrm{Hess}((f \circ R_\theta)(\xi)) - \mathrm{Hess}((f \circ R_\theta)(0_\theta))\| \leq \beta_{L2} \|\xi\|_\theta$$

Here, $\|\cdot\|$ on the left hand-side denotes the operator norm in $T_\theta\mathcal{M}$ given by

$$\|\mathrm{Hess}((f \circ R_\theta)(\xi))\| := \sup\{\|\mathrm{Hess}((f \circ R_\theta)(\xi))[\zeta]\|_{\theta^t} : \zeta \in T_\theta\mathcal{M}, \|\zeta\|_\theta = 1\}.$$

and $B_{\epsilon_1}(\theta^*) = \{\theta \in \mathcal{M} : \mathrm{dist}(\theta^*, \theta) < \epsilon_1\}$, $B_{\epsilon_2}(0_\theta) = \{\xi \in T_{\theta^*}\mathcal{M} : \|\xi\| < \epsilon_2\}$ the ball around θ^*, 0_θ with radius ϵ_1, ϵ_2, respectively.

1. *For all sequences $\{\theta^t\}_{t \in \mathbb{N}_0}$ generated by the algorithm converging to θ^*, there exists $T > 0$ such that there exists $c_d > 0$ such that*

$$\mathrm{dist}(\theta^{t+1}, \theta^*) \leq c_d \left(\mathrm{dist}(\theta^t, \theta^*)\right)^{\min\{\delta+1, 2\}},$$

   with $\delta > 0$ as in (2.23) *and for all $t > T$.*

2. *Further, if $\delta + 1 < 2$, then given $c_g > 1$, there exists $T > 0$ such that*

$$\|\mathrm{grad}\, f(\theta^{t+1})\|_{\theta^{t+1}} \leq c_g \|\mathrm{grad}\, f(\theta^t)\|_{\theta^t}{}^{\delta+1}$$

   for all $t > T$.

The variable δ in the exponential is the variable δ from equation (2.23)

Proof. We refer to [2, Section 7.4.2], [1] for a proof. □

2.2.5 Superlinear Methods based on Vector Transport

In the previous section, we introduced the Riemannian Newton trust-region method which can be seen as a practical Newton's method with a specific safeguarding strategy.

However, trust-region methods may not perform ideally on all problems and an expression for the Riemannian Hessian is not always available [2, Chapter 8]. In such a case, either a Hessian approximation (e.g. based on finite differences) is necessary or alternative algorithms with local superlinear convergence can be used. Typical algorithms with local superlinear convergence are the (nonlinear) conjugate gradient (CG) method and the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method [105, Section 5.2 & Section 8.1]. As these are line-search methods, one can transfer their concepts to Riemannian manifolds via a retraction (Section 2.2.2). However, for computing descent directions in the mentioned algorithms, the computation of the gradient at two different positions $\theta^{t+1}, \theta^t \in \mathcal{M}$ needs to be computed, that is the expression

$$\operatorname{grad} f(\theta^{t+1}) - \operatorname{grad} f(\theta^t). \tag{2.24}$$

Here, the gradient of θ^{t+1}, i.e. $\operatorname{grad} f(\theta^{t+1})$, is a tangent vector in the tangent space $T_{\theta^{t+1}}\mathcal{M}$ whereas the gradient of θ^t, i.e. $\operatorname{grad} f(\theta^t)$, is a tangent vector in the tangent space $T_{\theta^t}\mathcal{M}$, meaning that the two gradients possibly live in different spaces. Thus, in order to generalize the CG method and the BFGS method to a Riemannian setting, we need a tool that allows us to compare tangent vectors at distinct points on the manifold. For this reason, we introduce *vector transport* in the following.

Definition 2.33. (vector transport)*[120, Definition 4.1] We introduce the* Whitney sum *as*

$$T\mathcal{M} \oplus T\mathcal{M} \coloneqq \{(\eta, \xi) \mid \eta, \xi \in T_\theta\mathcal{M}, \theta \in \mathcal{M}\}.$$

A map $\mathscr{T} : T\mathcal{M} \oplus T\mathcal{M} \to T\mathcal{M} : (\eta, \xi) \mapsto \mathscr{T}_\eta(\xi)$ is called a vector transport *on $\mathcal{M}$ if there exists a retraction R on $\mathcal{M}$ and $\mathscr{T}$ satisfies the following conditions for any $\theta \in \mathcal{M}$:*

1. $\mathscr{T}_\eta(\xi) \in T_{R_\theta(\eta)}\mathcal{M}, \quad \eta, \xi \in T_\theta\mathcal{M},$
2. $\mathscr{T}_{0_\theta}(\xi) = \xi, \quad \xi \in T_\theta\mathcal{M},$
3. $\mathscr{T}_\eta(a\xi + b\zeta) = a\mathscr{T}_\eta(\xi) + b\mathscr{T}_\eta(\zeta), \quad a, b \in \mathbb{R}, \eta, \xi, \zeta \in T_\theta\mathcal{M}.$

Note that vector transport is not a standard concept of differential geometry [120, Section 4.3]. However, when the retraction R is the exponential map, we get the *parallel translation*, which is an important tool in classical differential geometry. The reason for using more general vector transports is that it is often computationally more efficient, similar to the motivation for using standard retractions instead of exponential maps. A vector transport can also be characterized by an affine connection and a retraction, for details see [2, Section 8.1.1] and [22, Section 10.3].

If $\mathcal{M}$ is an Euclidean submanifold endowed with a retraction R, we can find a vector transport in what follows: Since $T_\theta\mathcal{M} \subset \mathcal{E}$ for any $\theta \in \mathcal{M}$, we can define the vector transport $\mathscr{T}$ for $\eta, \xi \in T_\theta\mathcal{M}$ by

$$\mathscr{T}_\eta(\xi) = \mathrm{Proj}_{R_\theta(\eta)}(\xi), \tag{2.25}$$

where Proj_{R_θ} denotes the orthogonal projector from $\mathcal{E}$ to $T_{R_\theta(\eta)}$, see Section 2.1.7.

Equipped with a tool to subtract gradients from different tangent spaces to each other, we can generalize the conjugate gradients method and the (limited-memory) BFGS method to a Riemannian setting. We briefly state the Riemannian CG method (Algorithm 5) and the Riemannian BFGS method in the following. Since they are not as central as the Riemannian trust-region method to this thesis, we do not discuss choices of step lengths or convergence theory of these methods here but refer to [120, Chapter 4] and [2, Sections 8.2-8.3] for details.

Algorithm 5: Riemannian (nonlinear) conjugate gradients method [2, p. 182]

```
Input: objective $f$, vector transport $\mathscr{T}$ with associated retraction $R$, initial
       iterate $\theta^0 \in \mathcal{M}$
Output: sequence of parameters $\{\theta^t\}$
1 Set $\eta^0 = -\operatorname{grad} f(\theta^0)$
2 for $t = 0, 1, \dots$ do
3     Compute a suitable step size $\alpha^t$ and set
          $\theta^{t+1} = R_{\theta^t}(\alpha^t\eta^t);$
4     Compute $\beta^{t+1}$ and set
          $\eta^{t+1} = -\operatorname{grad} f(\theta^{t+1}) + \beta^{t+1}\mathscr{T}_{\alpha^t\eta^t}(\eta^t).$
5 end for
```

Another method with local superlinear convergence theory is the BFGS method which can be transferred to a manifold setting with the help of a vector transport, called Riemannian BFGS method (R-BFGS). Similar to the Euclidean case, one can also formulate a version of the R-BFGS algorithm via approximations of the Hessian inverse, see [115]. Further, the R-BFGS method can also be extended to a Riemannian limited-memory BFGS method (R-LBFGS), the extension is straightforward to the Euclidean case [105, Section 9.1]. In Algorithm 6, we present the general R-BFGS method. Note that it is crucial for the algorithm that we get a descent direction in every iteration, for this reason the Wolfe conditions need to be fulfilled (see [105, 65, 115] for a thorough explanation).

Algorithm 6: Riemannian BFGS method [115, 65]

```
  Input: objective $f$, vector transport $\mathscr{T}$ with associated retraction $R$, initial
         iterate $\theta^0 \in \mathcal{M}$, initial Hessian approximation $B_0$, gradient $\operatorname{grad} f$
  Output: sequence of parameters $\{\theta^t\}$
1 for $t = 0, 1, \dots$ do
2     Obtain $\eta^t \in T_{\theta^t}\mathcal{M}$ by solving $B^t \eta^t = -\operatorname{grad} f(\theta^t)$;
3     Obtain suitable step-length $\alpha^t$ such that the Wolfe conditions are fulfilled
      and set
          $$\theta^{t+1} = R_{\theta^t}(\alpha^t \eta^t);$$
4     Set $s^t = \mathscr{T}_{\alpha^t\eta^t}(\alpha^t\eta^t)$ and $y^t = \operatorname{grad} f(\theta^{t+1}) - \mathscr{T}_{\alpha^t\eta^t}(\operatorname{grad} f(\theta^t))$;
5     Define the linear operator $B^{t+1} : T_{\theta^{t+1}}\mathcal{M} \to T_{\theta^{t+1}}\mathcal{M}$ by
          $$B^{t+1}p = \tilde{B}^t p - \frac{\langle s^t, \tilde{B}^t p\rangle_{\theta^{t+1}}}{\langle s^t, \tilde{B}^t p\rangle_{\theta^{t+1}}} \tilde{B}^t s^t + \frac{\langle y^t, p\rangle_{\theta^{t+1}}}{\langle y^t, s^t p\rangle_{\theta^{t+1}}} y^t \quad \text{for all } p \in T_{\theta^{t+1}}\mathcal{M}$$
      with
          $$\tilde{B}^t = \mathscr{T}_{\alpha^t\eta^t} \circ B^t \circ \left(\mathscr{T}_{\alpha^t\eta^t}\right)^{-1}.$$
6 end for
```

In the following chapter, we consider maximum likelihood estimation, which is a typical approach for fitting parameters in statistical models. We will see later in this thesis that we can formulate maximum likelihood estimation problems involving covariance matrices as Riemannian optimization problems.

CHAPTER 3

Maximum Likelihood Estimation and the EM algorithm

Maximum likelihood estimation is a popular approach to make inference about the distribution of data arising from an experiment. We assume that the distribution is characterized by a parameter and we aim to find this parameter such that it fits observed data. The question we seek to answer with maximum likelihood estimation is *"under an assumed statistical parametric model, which parameters make the observed data most probable?"* [86]. Estimating parameters of statistical models results in the optimization of a (log-)likelihood and is the basis for the objectives studied in Chapter 5 and Chapter 6.

In this chapter, we review the method of maximum likelihood estimation for parameter estimation in statistical models. We put a special focus on estimation problems with unobservable information, i.e. we assume that there exist latent, unobservable variables. We first give a brief introduction to maximum likelihood estimation in Section 3.1. Difficult maximum likelihood estimation tasks frequently involve latent variables and are often solved by the Expectation Maximization (EM) algorithm. Due to its popularity and relevance to this thesis, we study the setting of maximum likelihood estimation with hidden information and present the EM algorithm with its properties in Section 3.2.

3.1 Maximum Likelihood Estimation for Parameterized Density Estimation

We start with introducing the statistical setting of interest. The notation mainly follows the introductory books [86, 128, 47].

We consider a *sample* denoted by $x = (x_1, \dots, x_n)$, $x_i \in \mathbb{R}^d$ that has been produced by an experiment and the central goal is to find the probability distribution F that the data $x_1, \dots, x_n$ is drawn from. We assume that the probability distribution function F admits a corresponding continuous probability density function f.

In the statistical literature, there are two major approaches to tackle the problem of fitting a probability distribution to known data: the *parametric* approach and the *nonparametric* approach [86]. In this thesis, we consider the parametric approach: We assume that the distribution is known up to values of a parameter θ from some finite-dimensional parameter space Θ. Here, the parameter space Θ can be identified with a subset of $\mathbb{R}^p$ for some $p \geq 1$ [86]. We summarize the notation used in this chapter in the following.

Notation:

- We assume that the observations $x = (x_1, \dots, x_n)$ are drawn from a random vector $X = (X_1, \dots, X_n)$. We denote the joint probability distribution function by F_θ. We write $X \sim F_\theta$.
- To avoid confusion, we write $x = (x_1, \dots, x_n)$ for real-valued observations and $x^v = (x^v_1, \dots, x^v_n)$ for real-valued variables arising in probability density functions. For fixed $\theta \in \Theta$, the expression
$$f_X(x; \theta)$$
denotes a real-valued scalar (namely f_X evaluated at observation x), whereas the expression
$$f_X(x^v; \theta)$$
denotes a function of the arguments x^v and θ.
- For each $\theta \in \Theta$, we suppose that F_θ admits a probability density function f_θ such that $F_\theta(x^v) = \int f_\theta(x^v) \,\mathrm{d}\, x^v$. In order to stress the dependence of the random variable X, we write $f_X(\,\cdot\,; \theta)$ for f_θ.

Our goal is to find values for θ such that f_θ is an appropriate model for the observed data. One way to achieve this is given by the *maximum likelihood approach*: the key idea consists in maximizing the joint probability of $X_1, \dots, X_n$ over the parameter space Θ, while fixing the random variables at the observation samples $x_1, \dots, x_n$ [86, Section 3.1.3].

We consider the joint distribution of $X_1, \dots, X_n$, that is

$$f_X(x^v; \theta) = f_{X_1,\dots,X_n}(x_1^v, \dots, x_n^v; \theta) \tag{3.1}$$

and fix it at the observations $x_1, \dots, x_n$. If the random variables $X_1, \dots, X_n$ are independent and identically distributed (iid) with probability density functions f_{X_i}, $i = 1, \dots, n$, the joint probability density function reads

$$f_X(x^v; \theta) = \prod_{i=1}^{n} f_{X_i}(x_i^v; \theta), \tag{3.2}$$

where $x^v = (x_1^v, \dots, x_n^v)$. When fixing the joint probability density function (3.1) at the observations $x_1, \dots, x_n$, we can express it as a function of θ, only. The procedure now consists of maximizing this function (3.1) over all possible values of θ: we seek to find the parameter $\hat{\theta}$ that makes the observed sample $x_1, \dots, x_n$ most probable under the model assumption f_θ [86, Section 3.1.3]. This leads to the following definitions.

Definition 3.1. (likelihood function, log-likelihood function, maximum likelihood estimator, maximum a posterior estimator) *[128, Chapter 2] We assume the setting introduced previously.*

1. *The* likelihood function *of $x = (x_1, \dots, x_n)$ is given by*

$$L(\theta) = f_X(x) = f_{X_1,\dots,X_n}(x_1, \dots, x_n; \theta).$$

2. *If the maximum of $L(\theta)$ over Θ exists, a* maximum likelihood estimator (MLE) *$\hat{\theta}$ of θ is given by*

$$\hat{\theta} = \arg\max_{\theta \in \Theta} L(\theta) \tag{3.3}$$

3. *The log-likelihood function of $x_1, \dots, x_n$ is given by*

$$l(\theta) = \log \left(f_{X_1,\dots,X_n}(x_1, \dots, x_n; \theta) \right).$$

Since the logarithm is a monotonically increasing function, it holds that

$$\hat{\theta} = \arg\max_{\theta \in \Theta} L(\theta) = \arg\max_{\theta \in \Theta} l(\theta).$$

4. *Similar to the frequentist setting above, we define a MLE for the Bayesian setting (see Appendix A.2). Assume we have a prior density for θ denoted by $g(\theta)$. Then, the* maximum a posterior estimator $\hat{\theta}_{MAP}$ *is defined as the maximizer of the posterior density of θ given $x_1, \cdots x_n$, that is [86, Chapter 8]*

$$\begin{aligned}\hat{\theta}_{MAP} &= \arg\max_{\theta\in\Theta} f_{X_1,\ldots,X_n}(x_1,\ldots,x_n;\theta)g(\theta)\\ &= \arg\max_{\theta\in\Theta} L(\theta)g(\theta).\end{aligned}$$

Considering the log-likelihood function l is particularly useful in the case of independent and identically distributed random variables. Then, the log-likelihood reads

$$l(\theta) = \sum_{i=1}^{n} \log f_{X_i}(x_i;\theta) \tag{3.4}$$

which usually yields objectives that are easier to differentiate [47, Section 8.2].

In the statistical literature, one often talks about "the" maximum likelihood estimator suggesting that it is unique. However, uniqueness of the solution might not be given although we obtain it in many classical probability distributions, where L is concave and the MLE can even be derived analytically. As an example, we show the MLE for the Gaussian distribution in the following. For more complicated probability distributions, there might exist many local maxima of the likelihood function [94, Section 3.4]. Another issue with the definition of the MLE is that L can be unbounded from above, such that existence of a maximum likelihood estimator is not ensured [47, Section 8.2]. An example for this is the maximum likelihood estimation of a univariate Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$, where the mean μ coincides with a single observation x_{i^*}; we refer to [79] for more examples. In such a case, an alternative can be to use penalized maximum likelihood estimation in order to obtain a *penalized maximum likelihood estimator* [94, Section 5.18].

The Gaussian distribution plays a central role in this thesis, for what reason we briefly present its MLE. Assume that $X_1,\ldots,X_n \overset{iid}{\sim} \mathcal{N}(\mu,\Sigma)$, where $\mathcal{N}(\mu,\Sigma)$ denotes the Gaussian distribution with mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$. The parameter θ and its according parameter space Θ are given by [128, Section 2.9.1]

$$\theta = (\mu,\Sigma), \qquad \Theta = \mathbb{R}^d \times \mathbb{P}^d.$$

Here, $\mathbb{P}^d$ denotes the set of symmetric, positive definite matrices of dimension d, that is

$$\mathbb{P}^d = \{A \in \mathbb{R}^{d\times d} : A = A^T, x^T A x > 0 \ \forall x \in \mathbb{R}^d \setminus \{0\}\} \tag{3.5}$$

The log-likelihood function of $x_1, \ldots, x_n$ reads

$$l(\theta) = -\frac{1}{2}\left(n(d\log(2\pi) + \log\det(\Sigma)) + \sum_{i=1}^{n}(x_i - \mu)^T\Sigma^{-1}(x_i - \mu)\right). \tag{3.6}$$

Since (3.6) is a differentiable and concave function, we get the maximum likelihood estimator $\hat{\theta}$ by setting the gradient of $l(\theta)$ equal to zero. We obtain the maximum likelihood estimator

$$\hat{\theta} = (\hat{\mu}, \hat{\Sigma}),$$

where

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})(x_i - \hat{\mu})^T.$$

3.2 Maximum Likelihood Estimation with Latent Variables

In some cases, the maximum likelihood estimator cannot be derived analytically and iterative algorithms become necessary. Nonlinear optimization algorithms can be used, but precaution is required that iterations are performed over the respective parameter space Θ [94, Section 1.3]. For example, in many statistical models we have a positive definiteness constraint which might be computationally expensive to impose (see Section 4.1). A popular alternative is to use the Expectation Maximization algorithm (EM) which is designed for settings where there is *unobservable information* or *hidden data.* We give a brief introduction to such a setting and derive the EM algorithm. The content that follows can be found in many books on computational statistics, for example [101, 19, 94], we here follow the notations from [101, Chapter 11] and [94].

As before, we consider realizations $x = (x_1, \ldots, x_n)$ of a random variable $X = (X_1, \ldots, X_n)$. Additionally to this, we assume that there exist so-called *hidden* or *latent* random variables $Z = (Z_1, \ldots, Z_n)$ which $X_1, \ldots, X_n$ depend on. Thus, the probability density function f_X is a marginal likelihood, that is we get f_X by integrating Z out of the joint distribution,

$$f_X(x^v; \theta) = \int_{-\infty}^{\infty} f_{(X,Z)}(x^v, z^v; \theta)dz^v.$$

Here, $f_{(X,Z)}$ denotes the joint distribution of X and Z. If we had complete data, that is if we had observations $z = (z_1, \dots, z_n)$ for the random variables $Z_1, \dots, Z_n$, the log-likelihood of X would read [94, Section 1.5.1]

$$l_X(\theta) = \log f_X(x;\theta) = \log \left(\int_{-\infty}^{\infty} f_{(X,Z)}(x, z; \theta) dz \right). \tag{3.7}$$

Unfortunately, the expression (3.7) is intractable because the variables $z_1, \dots, z_n$ are not observable, i.e. they are *latent*. Still, for many practical problems, the *complete* log-likelihood $l^c_{X,Z}$ of (X, Z) given by

$$l^c_{(X,Z)} = \log \left(f_{(X,Z)}(x, z; \theta) \right) \tag{3.8}$$

could be easily maximized *if* the z_i were observable. For many problems, the maximization of (3.8) would yield analytical solutions if the z_i were available. The widely used Expectation Maximization algorithm is an iterative algorithm that exploits this. Instead of considering the complete log-likelihood of X and Z which is unavailable, we take its expectation with respect to the latent variable Z. The algorithm then alternates between an *expectation step (E-step)*, where the expectation of the complete log-likelihood is computed, and a *maximization step (M-step)*, where the expression is maximized with respect to θ. We state it in the following.

3.2.1 Expectation Maximization Algorithm

The Expectation Maximization algorithm was introduced by Dempster, Laird and Ware [41] in 1977, although a similar idea has been proposed by earlier works [94, Section 1.8.2].

The Expectation Maximization algorithm alternates between the following two steps.

E-step:
At an iterate θ^t, $t = 0, 1, \dots$, we take the expectation of the logarithm of the joint log-likelihood of X and Z (called the *expected complete log-likelihood*). Here, the expectation is taken with respect to the conditional random variable $Z|X = x$ dependent of the current iterate θ^t, that is we compute the function

$$\begin{aligned} Q(\theta, \theta^t) &= \mathbb{E}_{Z=z^v|X=x;\theta^t} \left[\log \left(f_{(X,Z)} \left((x, z^v); \theta \right) \right) \right] \\ &= \int_{-\infty}^{\infty} \log \left(f_{(X,Z)}((x, z^v); \theta) \right) f_{Z=z^v|X=x} \left((z^v|x); \theta^t \right) dz^v. \end{aligned} \tag{3.9}$$

M-step:
Choose θ^{t+1} from the parameter space Θ such that the expectation of the complete log-likelihood (3.9) is maximized, that is

$$\theta^{t+1} = \arg\max_{\theta \in \Theta} Q(\theta, \theta^t). \tag{3.10}$$

We state the method in Algorithm 7.

Note that for many probabilistic models, the latent variables Z are assumed to be discrete, for which reason the evaluation of (3.9) consists of a sum instead of an integral (see Appendix B and [94]).

Algorithm 7: Expectation Maximization algorithm [19, p. 440f.]

```
  Input: initial parameter $\theta^0$ in parameter space $\Theta$
  Output: sequence of parameters $\{\theta^t\}$
1 for $t = 0, 1, 2, \dots$ do
2     E-step: Set
          $Q(\theta, \theta^t) = \mathbb{E}_{Z|X=x;\theta^t} \left[ \log \left( f_{(X,Z)}((x, z^v); \theta) \right) \right]$ ;
3     M-step: Maximize $Q(\theta, \theta^t)$ subject to $\theta \in \Theta$ and set
          $\theta^{t+1} = \arg\max_{\theta \in \Theta} Q(\theta, \theta^t)$;
4 end for
```

The main motivation for the EM algorithm is that maximization of the expected complete log-likelihood Q is much easier than the maximization of the original (marginal) log-likelihood $l_X(\theta)$. For many statistical models, the objective in the M-step (3.10) is concave and there exists a closed-form solution [101, Section 11.4]. In such cases, the per-iteration costs of Algorithm 7 are remarkably low. However, when the problem in the M-step is more complicated, iterative solvers for problem (3.10) become necessary. Then, we speak of *Generalized Expectation Maximization*, where a nonlinear optimizer is used for the M-step, or *Expectation Conditional Maximization*, where the problem is divided into several, easier constrained subproblems [19, p. 454], [94, Section 1.5]. Precaution is required that the problem in the M-step is well-defined, which is fulfilled in most application problems. What might be more challenging is that the parameter θ remains in its parameter space Θ. This might add an additional computational overhead, which weakens the simplicity of the EM algorithm [40, Section 2.14.2].

The EM algorithm can be interpreted as an alternating direction method [105, Section 2.7], where we alternate between maximization of the parameters of the distribution of the latent variables and maximization of the complete log-likelihood, see [58, Section 8.5.3].

A typical stopping criterion of Algorithm 7 is if the relative change in either the log-likelihood or in the parameter itself between two subsequent iterations falls below some threshold [94, Section 4.9]. Although this is in general rather a measure of lack of progress and not necessarily an indication of convergence, this stopping criterion is meaningful as it can be shown that the log-likelihood increases in every step of EM unless a critical point is reached (see [41] and explanation in Section 3.2.2).

The Expectation Maximization algorithm can also be modified for a Bayesian framework, namely for maximum a posterior (MAP) estimation with latent variables. For this, in the E-step, we compute

$$Q_{MAP}(\theta, \theta^t) = Q(\theta, \theta^t) + \log\left(g(\theta)\right), \tag{3.11}$$

where $Q(\theta, \theta^t)$ is as in (3.9) and $g(\theta)$ is a suitable prior density for θ. In the M-step, the conditional expectation of the log-complete data posterior density (3.11) is maximized, see [94, Section 1.6.1]. The properties of EM discussed in the next section generalize also to the Bayesian version of EM for maximum a posterior estimation.

3.2.2 Properties of the Expectation Maximization Algorithm

At a first glance, the approach of considering the complete log-likelihood instead of marginalizing the hidden data out by equation (3.7) is not obvious since we cannot simply interchange the logarithm and the integral. We will now outline why this approach is meaningful nevertheless.
As the logarithm is a concave function, we can apply Jensen's inequality and get the lower bound

$$l_X(\theta) \geq Q(\theta, \theta^t) - \mathbb{E}_{Z|X=x}\left[\log\left(f_{Z|X=x}\left((z^v|x); \theta^t\right)\right)\right] \tag{3.12}$$

for all $t = 0, 1, \ldots$. The expected complete log-likelihood is a lower bound for the log-likelihood of X. Further, the lower bound in (3.12) for $\theta = \theta^t$ reads

$$Q(\theta^t, \theta^t) - \mathbb{E}_{Z|X=x}\left[\log\left(f_{Z|X=x}\left((z^v|x); \theta^t\right)\right)\right] = \log\left(f_X(x; \theta^t)\right) = l_X(\theta^t)$$

where Bayes' rule is used, see Appendix A.2. Thus, the lower bound in (3.12) coincides with the log-likelihood we aim to maximize at the point θ^t. This means that the lower bound is tight after the E-step and maximizing the expected complete log-likelihood $Q(\theta, \theta^t)$ will lift the log-likelihood $l(\theta)$ to a higher level. In total, we have

$$l_X(\theta^{t+1}) \geq Q(\theta^{t+1}, \theta^t) \geq Q(\theta^t, \theta^t) = l_X(\theta^t), \tag{3.13}$$

where the second inequality follows from the maximization step (3.10). The Expectation Maximization algorithm, Algorithm 7, thus ensures that the log-likelihood is non-decreasing in every iteration. We have equality in (3.13) if θ^t is a stationary point and l_X is bounded from above [101, Section 11.4].

Besides its properties of improving the log-likelihood in every step, the (generalized) EM algorithm converges to a stationary point under suitable conditions. Details can be found in the original work on the Expectation Maximization algorithm by Dempster, Laird and Rubin [41], who derive convergence results to local maxima lying in the interior of the parameter space Θ. The work by Nettleton [103] transfers the conditions of [41] to cases where a local maximum lies at the boundary of the parameter space Θ. Although EM theoretically can converge to stationary points that are not local maxima, such cases are of rather artificial nature and rarely occur in practice. We refer to [94, Section 3.6] for examples of such unnatural behavior. A remedy to this can be to restart the algorithm from another value [94, Section 3.6].

The convergence rate of EM has been shown to be linear [41] and for some examples, the convergence rate can be worse, namely sublinear [63]. The slow convergence of EM is a well-known drawback addressed in many earlier works [145, 88] and more recent works [31, 146, 117]. The convergence rate of EM is related to the share of hidden, unobservable information in the data: a higher share usually results in (sub-) linear convergence [94, Section 3.9.3], [133]. On the other hand, Salakhutdinov, Roweis and Ghahramani [117] showed that EM has local superlinear behavior similar to Newton's method if the ratio of missing information to complete information is very small. However, this is rarely the case in practice such that we mostly end up in linear convergence [94, Section 3.9].

Since the famous work by Dempster, Laird and Rubin [41], some approaches have been proposed to speed up the EM algorithm. These approaches mostly rely on hybrid approaches between EM and nonlinear optimization techniques like the conjugate gradient method (hybrid EM-ECG algorithm [117], generalized conjugate gradients algorithm [71]) or Newton's method [4, 116]. A detailed discussion of these extensions can be found in [94, Chapter 4]. Another approach is given by adapting the M-step such that maximization is performed over a larger, statistically meaningful parameter space. This approach is known as parameter-expanded EM, we refer to [88, 83] for a thorough analysis. Although these methods often lead to better convergence rates, their use in practice is often limited and the standard EM algorithm (Algorithm 7) is used. This is because the mentioned extensions of EM usually add a big computational overhead by imposing additional constraints such as positive definiteness of covariance matrices, resulting in higher per-iteration costs and thus weakening the advantage of an improved convergence rate [94, Chapter 4]. In addition, most hybrid methods do not guarantee any more that the likelihood increases (or remains the same in the optimum) in every step, which is a very strong property of Algorithm 7. Besides, theoretical results for the convergence speed for some extensions of EM are missing and only experimental evidence is available. Thus, the Expectation Maximization in its standard form is often used in practice despite its low convergence. Its simplicity, its non-decreasing property in every step and its statistical interpretation are the main advantages which make it favored for users.

The Expectation Maximization algorithm is a popular choice when it comes to maximum likelihood estimation with covariance matrices, because the M-step then often admits a closed-form solution such that the positive definiteness constraint is automatically fulfilled. In Chapter 4, we review alternative numerical methods to estimate covariance matrices and discuss their properties. Motivated by the drawbacks of existing methods, we study the Riemannian geometry of covariance matrices, allowing to make use of the concepts of Riemannian Optimization introduced in Chapter 2.

CHAPTER 4

Riemannian Geometry of Covariance Matrices

In the previous chapter, we introduced maximum likelihood estimation to fit parameterized statistical models. For many models, this results in optimization problems over covariance matrices. Thus, we often need to impose positive definiteness of matrices as a constraint. This chapter deals with optimization with respect to covariance matrices and is divided into two parts. In Section 4.1, we briefly review approaches for nonlinear optimization with respect to covariance matrices. Motivated by the limitations of these approaches, the second part, Section 4.2, deals with the Riemannian geometry of positive definite matrices. This allows to formulate challenging maximum likelihood estimation tasks as Riemannian optimization problems over product manifolds incorporating the manifold of positive definite matrices. The present chapter thus connects the two previous chapters and builds the basis to use Riemannian optimization for the two statistical models studied in Chapter 5 and 6.

4.1 Estimation Approaches for Covariance Matrices in Statistical Models

For many probability distributions, the parameter of interest θ in maximum likelihood estimation involves the covariance matrix denoted by Σ. An important property of a covariance matrix Σ is that it is positive semidefinite [48, Chapter 5]. However, for many distributions such as the Gaussian distribution, the condition is even stronger, that is the covariance matrix must be (strictly) positive definite in order to have well-defined expressions for the probability distribution and density functions [86, Section 3.7]. For MLE, this means maximization over coordinates of the parameter space $\mathbb{P}^d$, where $\mathbb{P}^d$ denotes the set of positive definite matrices characterized by

$$\mathbb{P}^d = \{\Sigma \in \mathbb{R}^{d\times d} | \Sigma = \Sigma^T \text{ and } \Sigma \succ 0\} \tag{4.1}$$

see (3.5). Here, the notation $\Sigma \succ 0$ means that Σ is positive definite, that is $x^T\Sigma x > 0$ for all $x \in \mathbb{R}^d \setminus \{0\}$.
Riemannian optimization has recently gained increasing interest in the field of data science If the problem can be formulated as a problem with hidden information like described in Section 3.2, the Expectation Maximization algorithm is often the method of choice [94]. The advantage of using Expectation Maximization for MLE involving covariance matrices is that Algorithm 7 produces iterates that fulfill positive semidefiniteness, and in most applications strict positive definiteness with the EM algorithm is fulfilled [94, Section 2.7]. However, as pointed out in Section 3.2.2, the EM algorithm suffers from slow convergence, especially if the share of hidden information is high [94, Section 3.9.3]. Alternatively, Newton-type methods might be a better choice in order to achieve superlinear convergence [105]. Still, when it comes to optimizing over the open cone (4.1), precaution is required that iterates stay in the interior of the set $\mathbb{P}^d$.

One way to ensure this is to reformulate the objective with a Cholesky decomposition $\Sigma = LL^T$ and to impose positivity of the diagonal elements of the lower triangular matrix $L \in \mathbb{R}^{d\times d}$ [15]. This approach consists of optimizing over $d(d+1)/2$ entries of the lower triangular matrix L with additional constraints. Though, when d increases, the number of parameters to be estimated increases quadratically, making the approach inappropriate for larger dimensions. In addition, such a reformulation can add additional spurious critical points [65, 24].

A related field to satisfy at least a positive semidefinite condition is the field of semidefinite programming [143]. Here, the condition $\Sigma \succeq 0$ is ensured by a reformulation as a Cholesky decomposition, for example by using smooth convex inequalities [139] or by a mapping to a nonlinear optimization problem by projections [24]. Most algorithms for semidefinite programming are based on interior point methods [143]. Nevertheless, semidefinite programming is related to convex or even linear optimization problems, which is rarely the setting for challenging maximum likelihood estimation tasks. Extensions to non-convex objectives are available and based on convex relaxations [49, 107, 92], but semidefinite programming for convex functions can already be challenging and the relaxation should be tight in order to get meaningful results [13]. Another issue with methods for semidefinite programming is the limited scalability to larger problems. In a recent work, Majumdar et. al [92] surveyed scalability advances in semidefinite programming, like exploiting sparsity. But such extensions are introduced for linear or convex problems, making them hardly usable for the problem of complicated maximum likelihood estimation.

As outlined above, existing methods for maximum likelihood estimation for covariances based on Cholesky decompositions come with some drawbacks, for what reason in practice often the Expectation Maximization algorithm is used despite the slow convergence rate [101, Section 11.4]. Recently, exploiting the geometry of positive definite matrices to apply Riemannian optimization has gained increasing interest and often allows for faster convergence. In the last decade, there has been a lot of research on the geometry of positive definite matrices in the field of machine learning. This is mainly motivated by a wide range of application fields raising from computer vision [29], (medical) image analysis [62, 99] to radar signal processing [9, 67]. For the purpose of Riemannian optimization, acting on the manifold of $\mathbb{P}^d$ is still quite novel. A prominent example of Riemannian optimization of $\mathbb{P}^d$ is the computation of the Karcher mean [73], that is the computation of the center of given positive definite matrices. The works on fitting Gaussian mixture models with a Riemannian approach by Hosseini and Sra [65, 66] recently have gained attention and are also the basis for Chapter 5 of this thesis. The former work by Sra and Hosseini, appearing as [130] in the bibliography, derives important results on geodesic convexity on the manifold of positive definite matrices with a special focus on maximum likelihood estimation for elliptically contoured distributions.

In the following, we review the Riemannian geometry of positive definite matrices to make use of the concepts of Riemannian optimization introduced in Chapter 2. Equipped with the Riemannian tools for the parameter space of interest, we are then able to apply Riemannian optimization on two maximum likelihood estimation tasks in Chapter 5 and Chapter 6.

4.2 The Manifold of Positive Definite Matrices

The set of symmetric positive definite matrices is well-known to be a smooth manifold [17, Chapter 6], [98]. In this section, we review the differential-geometric properties of the set of symmetric positive definite matrices needed to perform Riemannian optimization on that space.

We recall that the parameter space we seek to optimize over is the set of (symmetric) real positive definite matrices of fixed dimension d denoted by

$$\mathbb{P}^d = \{\Sigma \in \mathbb{R}^{d \times d} | \Sigma = \Sigma^T \text{ and } \Sigma \succ 0\}.$$

The set $\mathbb{P}^d$ is an open cone in the set of symmetric real matrices of size d [17, Chapter 6], denoted by

$$\mathbb{S}^d = \{A \in \mathbb{R}^{d \times d} | A = A^T\}. \tag{4.2}$$

As an open set in an Euclidean space, it is an Euclidean submanifold in the sense of Definition 2.18 and Theorem 2.17 with dimension $d(d+1)/2$. Further, $\mathbb{P}^d$ is an open submanifold, hence its tangent space is the whole embedding space: at a point $\Sigma \in \mathbb{P}^d$, the tangent space is given by

$$T_\Sigma \mathbb{P}^d = \mathbb{S}^d, \tag{4.3}$$

see Section 2.1.7. The manifold of positive definite matrices $\mathbb{P}^d$ can thus be globally approximated by the same linear space $\mathbb{S}^d$. In the following, we omit the index Σ for tangent vectors and simply write $\xi, \chi \in \mathbb{S}^d$ for tangent vectors.

4.2.1 Riemannian Metrics for Covariance Matrices

To make the manifold $\mathbb{P}^d$ a Riemannian manifold, we must equip its tangent space $\mathbb{S}^d$ with a suitable metric $\langle \cdot, \cdot \rangle_\Sigma$ at any positive definite matrix $\Sigma \in \mathbb{P}^d$. There are many metrics proposed for the manifold of positive definite matrices [134]. We here discuss three metrics that mostly appear in the context of Riemannian optimization on $\mathbb{P}^d$. They come with computationally tractable and efficient expressions for gradients, Hessians and retractions which is desirable for Riemannian optimization.

The most obvious idea of defining a Riemannian metric consists in inheriting the standard Euclidean inner product of the set of symmetric matrices, that is

$$\langle \xi, \chi \rangle_\Sigma^{sym} = \operatorname{tr}(\xi \chi), \tag{4.4}$$

where $\operatorname{tr}(A) = \sum_{i=1}^{d} a_{ii}$ is the trace of matrix A and $\xi,\, \chi \in \mathbb{S}^d$.

However, this inner product does not take into account the curvature of the set of positive definite matrices because it is independent of a point Σ on the manifold. Another issue with the Euclidean inner product (4.4) is that the associated geodesic $\gamma^{sym}_{\Sigma_1,\Sigma_2}$ joining two points Σ_1, Σ_2 in $\mathbb{P}^d$ is a straight line, that is [73]

$$\gamma^{sym}_{\Sigma_1,\Sigma_2}(t) = \Sigma_1 + t(\Sigma_2 - \Sigma_1). \tag{4.5}$$

This is unfavorable, since it might occur that for some choices of t, Σ_1, Σ_2, the matrix $\gamma^{sym}_{\Sigma_1,\Sigma_2}(t)$ is not positive definite [73, 62]. Furthermore, the geodesic (4.4) completely neglects the interpretation of $\mathbb{P}^d$ as a manifold. In Figure 4.1, the shortest path with respect to different metrics is visualized. We observe from the figure that the Euclidean metric $\langle \xi, \chi \rangle^{sym}_{\Sigma}$ neglects the curvature of the space $\mathbb{P}^d$ as an open cone (red, solid line). In contrast, the shortest paths induced by the affine-invariant metric and the Bures-Wasserstein metric are in line with the curvature property of $\mathbb{P}^d$, these are introduced in the following.

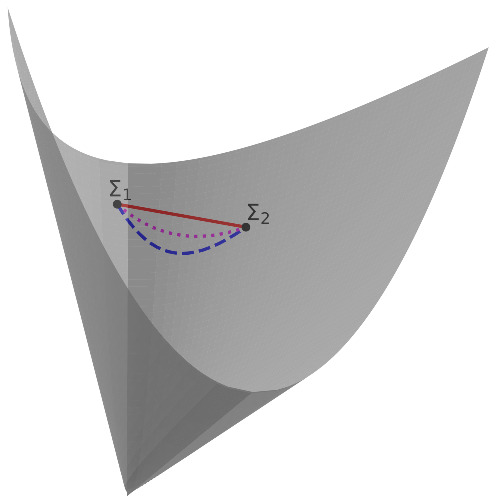


Figure 4.1: Open cone of $\mathbb{P}^2$. Visualized are the geodesics connecting $\Sigma_1, \Sigma_2 \in \mathbb{P}^2$ with respect to the Euclidean metric $\langle \cdot, \cdot \rangle^{sym}$ (red, solid line), the affine-invariant metric $\langle \cdot, \cdot \rangle^{ai}$ (blue, dashed line) and the Bures-Wasserstein metric $\langle \cdot, \cdot \rangle^{bw}$ (magenta, dotted line). Figure inspired by [62].

There are several other inner products that have been studied in the context of positive definite matrices. These slightly vary with points Σ on the manifold, which is the natural intuition of a *Riemannian* metric. In the context of Riemannian Optimization, the *affine-invariant metric* is the one that is most associated with the manifold of interest, see [99, 65, 148, 130]. Up to date, it is the default inner product implemented in toolboxes on manifold optimization (`Manopt` [23], `Manopt.jl` [16], `pymanopt` [136], `ROPTLIB` [68]) .

The affine-invariant metric is given by [17, Chapter 6]

$$\langle \xi_\Sigma, \chi_\Sigma \rangle_\Sigma^{ai} = \operatorname{tr}(\xi_\Sigma \Sigma^{-1} \chi_\Sigma \Sigma^{-1}). \tag{4.6}$$

This inner product is intuitively well suited for performing optimization on $\mathbb{P}^d$: close to the boundary of the parameter space, steps become large and we potentially leave the boundary fast. The matrix exponential and matrix logarithm play a central role for the affine-invariant geometry. For completeness reasons, we state them in the following. The matrix exponential for a non-singular matrix A is given by the convergent series

$$\exp(A) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k.$$

Accordingly, logarithms of a matrix A are solutions X to the matrix equation $\exp(X) = A$. If A has positive eigenvalues, there exists a unique real logarithm denoted by $\log(A)$, the *matrix logarithm*, see [36, 98].

The Riemannian distance under the affine-inner product is given by

$$\operatorname{dist}^{ai}(\Sigma_1, \Sigma_2) = \|\log\left({\Sigma_1}^{-1/2} \Sigma_2 {\Sigma_1}^{-1/2}\right)\|_F, \tag{4.7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\log(\cdot)$ the (unique) matrix logarithm [17, Section 6.1]. This distance is motivated by the hyperbolic distance between two positive scalars [98]: One can show that

$$\operatorname{dist}^{ai}(\Sigma_1, \Sigma_2) \geq \|\log(\Sigma_1) - \log(\Sigma_2)\|_F, \tag{4.8}$$

for all $\Sigma_1, \Sigma_2 \in \mathbb{P}^d$, so the map $(\mathbb{S}^d, \|\cdot\|_F) \xrightarrow{\exp} (\mathbb{P}^d, \operatorname{dist}^{ai})$ increases distances or is *metric-increasing* [17, Theorem 6.1.4]. In case of commuting matrices ξ_Σ, χ_Σ, there is equality in (4.8) and the geodesic distance thus equals the Euclidean distance between the matrix logarithms [17, Section 6.1]. Further, there exists a unique geodesic that realizes dist^{ai} which is given by [17, Theorem 6.1.6]

$$\gamma^{ai}_{\Sigma_1,\Sigma_2}(t) = {\Sigma_1}^{1/2} \left({\Sigma_1}^{-1/2} \Sigma_2 {\Sigma_1}^{-1/2}\right)^t {\Sigma_1}^{1/2}, \qquad \text{for } 0 \leq t \leq 1. \tag{4.9}$$

It can be shown that the midpoint of the geodesic (4.9) is the *geometric mean* of two positive definite matrices [73], [17, Section 6.3], which is in line with the intuition of a shortest path between two points on a manifold.

Although the computational effort of evaluating the affine-invariant metric (4.6) is higher compared to the Euclidean inner product (4.4), Riemannian optimizers with the affine-invariant metric are expected to converge faster as they take the intrinsic geometry of $\mathbb{P}^d$ into account. A study for computing the geometric mean in [73] underlines this intuition.

The *Bures-Wasserstein metric* is another prominent metric that is often associated with covariance matrices. It is widely used in the context of information geometry and is used in applications like optimal transport [7, 72, 18, 93] and quantum information [127, 89]. The Bures-Wasserstein metric has a information-geometric interpretation: in the case of centered Gaussians, the two distances induced by the respective metrics coincide [18, 93]. We introduce the Bures-Wasserstein metric by

$$\langle \xi_\Sigma, \chi_\Sigma \rangle_\Sigma^{bw} = \frac{1}{2} \operatorname{tr}(\mathcal{L}_\Sigma[\xi_\Sigma]\chi_\Sigma) \tag{4.10}$$

where $\mathcal{L}_\Sigma[\xi_\Sigma]$ is the solution to the linear system $\mathcal{L}_\Sigma[\xi_\Sigma]\Sigma + \Sigma\mathcal{L}_\Sigma[\xi_\Sigma] = \xi_\Sigma$ and is known as the Lyapunov operator [57, 137]. The authors of [57] compared the widely used affine-invariant metric against the Bures-Wasserstein metric in the context of Riemannian optimization. The authors point out that the condition number of the Hessian at a point Σ^* with respect to the affine-invariant metric depends quadratically of Σ^*, whereas the condition number with respect to the Bures-Wasserstein metric depends linearly of Σ^*. This has an impact on the convergence speed of Riemannian optimization algorithms under the relying metric, especially when the condition number of Σ^* is large: the authors observed that for many objectives relevant to machine learning, the Bures-Wasserstein metric leads to faster algorithms. Yet, the authors show that maximization of the function

$$f(\Sigma) = \log\det(\Sigma) \tag{4.11}$$

yields a smaller condition number with the affine-invariant metric than with the Bures-Wasserstein metric. As the maximization of the logdet (4.11) plays a central role in the objectives studied in this thesis, we thus consider the affine-invariant metric in the following, only. Furthermore, throughout this thesis, we drop the superscript *ai* in the metric and define

$$\langle \xi_\Sigma, \chi_\Sigma \rangle_\Sigma := \langle \xi_\Sigma, \chi_\Sigma \rangle_\Sigma^{ai} = \operatorname{tr}(\xi_\Sigma \Sigma^{-1} \chi_\Sigma \Sigma^{-1}).$$

Other metrics than the ones discussed above include the log-Euclidean metric [69, 10], the log-det metric [30, 129, 26], the log-Cholesky metric [85] and the the Bogoliubov-Kubo-Mori metric [97]. In a recent work, the differential-geometric relationship between some of these metrics has been studied [134]. A detailed comparison of the different metrics is beyond the scope of this thesis and the understanding of metrics defined on $\mathbb{P}^d$ is still subject of current research [134, 135].

4.2.2 Riemannian Gradient and Riemannian Hessian for the Affine-invariant Metric

As outlined above, we will consider the affine-invariant metric (4.6) from now on denoted by $\langle \cdot, \cdot \rangle_\Sigma$. Let $f : \mathbb{P}^d \to \mathbb{R}$ be a smooth function and $\Sigma \in \mathbb{P}^d$, $\xi \in \mathbb{S}^d$. We denote the Riemannian gradient with respect to the metric (4.6) by $\operatorname{grad} f$. As $\mathbb{P}^d$ is a Euclidean submanifold, according to (2.7) we get

$$\operatorname{tr}(\xi \Sigma^{-1} \operatorname{grad} f(\Sigma) \Sigma^{-1}) = \operatorname{tr}(\xi \operatorname{grad}^{sym} \bar{f}(\Sigma))$$

where $\operatorname{grad}^{sym} \bar{f}$ denotes the gradient of the smooth extension $\bar{f}$ to the set of symmetric matrices with the Euclidean metric (4.4). Thus, the Riemannian gradient reads

$$\operatorname{grad} f(\Sigma) = \Sigma \left(\operatorname{grad}^{sym} \bar{f}(\Sigma)\right) \Sigma = \frac{1}{2} \Sigma \left(\operatorname{grad}^{e} \bar{f}(\Sigma) + \left(\operatorname{grad}^{e} \bar{f}(\Sigma)\right)^T\right) \Sigma, \tag{4.12}$$

where $\operatorname{grad}^e \bar{f}$ denotes the classical Euclidean gradient of the smooth extension $\bar{f}$ to the set of real matrices $\mathbb{R}^{d \times d}$ with the Euclidean metric $\langle A, B \rangle = \operatorname{tr}(A^T B)$ for $A, B \in \mathbb{R}^{d \times d}$ [73].

For the Riemannian Hessian, we specify the Riemannian connection ∇^{pd}. At a point $\Sigma \in \mathbb{P}^d$, it is given by

$$\nabla^{pd}_{V_\Sigma} U_\Sigma = \mathrm{D}\left(V_\Sigma\right)\left[U_\Sigma\right] - \frac{1}{2}\left(U_\Sigma \Sigma^{-1} V_\Sigma + V_\Sigma \Sigma^{-1} U_\Sigma\right), \tag{4.13}$$

where $U_\Sigma, V_\Sigma \in \mathfrak{X}(\Sigma)$ are vector fields whose domains include Σ and $\mathrm{D}(V)[U]$ is the classical Euclidean derivative of V along direction U [73]. It can be shown that the expression in (4.13) fulfills the Koszul formula stated in (2.3) and thus is uniquely identified with the Riemannian connection for the affine-invariant metric (4.6), see [109, 73]. Thus, the Riemannian Hessian at Σ reads

$$\begin{aligned} \operatorname{Hess} f(\Sigma)[\xi_\Sigma] &= \nabla^{pd}_{\xi_\Sigma} \operatorname{grad} f(\Sigma) \\ &= \mathrm{D}\left(\operatorname{grad} f(\Sigma)\right)\left[\xi_\Sigma\right] - \frac{1}{2}\left(\xi_\Sigma \Sigma^{-1} \operatorname{grad} f(\Sigma) + \operatorname{grad} f(\Sigma) \Sigma^{-1} \xi_\Sigma\right), \end{aligned} \tag{4.14}$$

where $\xi_\Sigma \in \mathbb{S}^d$ [73].

4.2.3 Retractions for the Affine-invariant Metric

A natural candidate for the Riemannian connection with respect to the affine-invariant metric (4.6) is given by the exponential map. The geodesic (4.9) can be equally expressed as

$$\gamma^{ai}_{\Sigma_1,\Sigma_2}(t) = \Sigma_1^{1/2} \exp\left(t \log\left({\Sigma_1}^{-1/2}\Sigma_2{\Sigma_1}^{-1/2}\right)\right) \Sigma_1^{1/2}$$

yielding the tangent vector

$$\dot{\gamma}^{ai}_{\Sigma_1,\Sigma_2}(0) = \Sigma_1^{1/2} \log\left({\Sigma_1}^{-1/2}\Sigma_2{\Sigma_1}^{-1/2}\right) \Sigma_1^{1/2}.$$

Thus, the exponential map Exp_Σ at $\xi \in T_\Sigma\mathbb{P}^d$ is given by

$$\mathrm{Exp}_\Sigma(\xi) = \Sigma^{1/2} \exp\left(\Sigma^{-1/2}\xi\Sigma^{-1/2}\right) \Sigma^{1/2}, \tag{4.15}$$

see [73, 17]. A computationally more efficient expression of the exponential map (4.15) is given by

$$\mathrm{Exp}_\Sigma(\xi) = \Sigma \exp(\Sigma^{-1}\xi), \tag{4.16}$$

which can be easily seen by the definition of the matrix exponential [130]. The use of the exponential map for Riemannian optimization methods is desirable especially for Newton-type algorithms because it is second-order (see Section 2.2.3). At the same time, evaluating the exponential map in every iteration usually comes with a high computational effort. In the case of the evaluation of the exponential map (4.16), we observe that the expression "$\Sigma^{-1}\xi$" must be evaluated in every iteration anyway because it appears in the inner product (4.6). Thus, the computational burden of (4.16) is determined by the computation of the matrix exponential. The Pade approximation [5] is an efficient way to compute the matrix exponential and is implemented in the toolboxes for Riemannian optimization [136, 23, 96]. In a study on computing the Karcher mean of positive definite matrices, the second-order approximation to the exponential map given by

$$R^{approx}_\Sigma(\xi) = \Sigma + \xi + \frac{1}{2}\xi\Sigma^{-1}\xi$$

was investigated [73]. However, the authors observed that the use of different retractions did not have a relevant impact on the performance of different Riemannian optimization algorithms. For this reason, we focus on the exact exponential map (4.16) throughout this thesis.

4.2.4 Vector transport for the Affine-invariant Metric

In order to apply specific quasi-Newton methods on Riemannian manifolds or use finite-difference approximations, one needs to specify a vector transport to subtract tangent vectors of different tangent spaces from each other, see Section 2.2.5.

A natural candidate for a vector transport for $\mathbb{P}^d$ is the one that is associated with the exponential map (4.16). It is given by [73, 65]

$$\mathscr{T}_\eta^\Sigma(\xi) = \Sigma^{\frac{1}{2}} \exp\left(\frac{1}{2}\Sigma^{-\frac{1}{2}}\eta\Sigma^{-\frac{1}{2}}\right) \Sigma^{-\frac{1}{2}}\eta\Sigma^{-\frac{1}{2}} \exp\left(\frac{1}{2}\Sigma^{-\frac{1}{2}}\eta\Sigma^{-\frac{1}{2}}\right) \Sigma^{\frac{1}{2}}, \tag{4.17}$$

where $\mathscr{T}_\eta^\Sigma(\xi)$ denotes the vector transport of $\xi \in T_\Sigma\mathbb{P}^d$ to $\eta \in T_\Sigma\mathbb{P}^d$.

The vector transport in (4.17) is computationally expensive which weakens the power of quasi-Newton methods for the manifold of positive definite matrices [51]. In the toolkits for Riemannian optimization `pymanopt` [136] and `Manopt` [23], the vector transport (4.17) is not implemented as a default. Instead, the identity mapping is used as a vector transport. The reasoning for this is that if the step-lengths performed by the optimizers are small enough, the transfer of gradients onto other tangent spaces can be neglected because the set of positive definite matrices is an open cone. However, such an approach neglects the curvature of $\mathbb{P}^d$ and might yield slower algorithms. In a recently published article, the authors Godaz et. al [51] proposed two mappings in the tangent space via a Cholesky decomposition and a second inverse root. This yields an associated vector transport equal to identity, thus avoiding the high computational costs of the vector transport (4.17). Nevertheless, the convergence speed for the vector transport based method (4.17) was comparable to their introduced *vector transport free* method.

In this chapter, we derived the Riemannian characteristics of positive definite matrices. Based on that, we are able to study Riemannian optimization for two challenging maximum likelihood estimation tasks involving the fitting of covariance matrices in the subsequent two chapters. In the following chapter, we study the *Gaussian mixture model.*

CHAPTER 5

Riemannian Optimization for Fitting Gaussian Mixture Models

Data clustering is an important field for machine learning thanks to its wide use in real-world applications in the absence of specific labeled information [3, Section 1.1]. In practice, labeled data is often expensive to acquire and sometimes contains relatively little information. For this reason, the demand for *unsupervised learning models* like clustering approaches is high [101, Section 1.3]. The general question addressed with data clustering can be posed as *"Given a set of data points, what is a good partition into groups such that data within a group are as similar as possible?"* [3, p. 2]. Due to the importance of clustering for data science, many models exist in the literature to address this problem [3], [101, Section 1.3]. A popular approach is given by *probabilistic model-based clustering* which seeks to optimize the fit between the observed data and some probabilistic model [3, Section 3.1]. Typically, the probabilistic model of choice is a *mixture model.* Here, we assume that the observed data are generated by a mixture of K underlying probability distributions [3, Section 3.1]. Such models allow to get a membership probability of each data point for each of the K components. Similarity within a cluster is characterized by data points sharing the same probability distribution [101, Chapter 11]. Mixture models are *soft clustering models* as the observations are not assigned to exactly one cluster but are assigned to a specific cluster with some individual probability [3, Section 1.2.2]. When observations are continuous-valued, the *Gaussian mixture model* is a popular choice for probabilistic model-based clustering. Here, we assume that each of the K components is a Gaussian distribution [101, Section 11.2.1]. Fitting the clustering problem by a Gaussian mixture model is thus transformed into a

parameter estimation problem as we need to specify the mixing proportions of the components and the parameters of the K Gaussian distributions [65]. Typically, parameter estimation is performed by a maximum likelihood approach [101, Section 11.2.1].
Besides their popularity for clustering tasks, Gaussian mixture models are also widely used as probability density estimators or approximators [52]. Since the Gaussian mixture model is a probability density function, we can approximate a continuous probability density function by a Gaussian mixture model with sufficient components [52], [101, Chapter 11].

Due to their generative modeling power, Gaussian mixture models are widely used in many different application fields like image analysis [6, 45, 149], pattern recognition [144, 19], econometrics [12, 33] and many others. The crucial part consists of finding the distribution of the K Gaussian components and the mixing proportions. The present chapter deals with this problem of parameter estimation in Gaussian mixture models.

The chapter is organized as follows. In Section 5.1, we formally introduce the Gaussian mixture model and the resulting optimization problem for estimating its parameters. We give a brief overview of optimization approaches for fitting Gaussian mixture models. In Section 5.2, we formulate the optimization problem as a Riemannian optimization problem according to Chapter 2. To that end, we use modifications for the objective as proposed in the works of Hosseini and Sra [65, 66] that make Riemannian optimization for Gaussian mixture models more tractable in Section 5.2.1. Based on this reformulation, we present the resulting Riemannian optimization problem in Section 5.2.2 and we derive expressions for the Riemannian gradient and the Riemannian Hessian. In Section 5.3, we introduce the Riemannian Newton trust-region algorithm for Gaussian mixture models and study its convergence. Subsequently, we show numerical results for the Riemannian Newton trust-region algorithm for Gaussian mixture models for both a clustering task and a probability density approximation task in Section 5.4. In the last part of this chapter, Section 5.5, we summarize our findings and give an outlook for potential future research directions.

The derivation of the Riemannian Hessian (Section 5.2.2) and the results of numerical experiments (Section 5.4) have been published in a refereed publication which appears as [124] in the bibliography.

5.1 Problem Formulation and Related Work

We introduce the Gaussian mixture model formally in the following. We follow the notations from [101, Chapter 11].

Let $X \in \mathbb{R}^d$ be a d-dimensional random variable and an integer $K \in \mathbb{N}$ be given. We assume that X follows a distribution whose probability density function p_X is a linear combination of K Gaussian distributions. Such a probability density function is called a *Gaussian mixture model* with K components. Formally, the Gaussian mixture model is given by the probability density function

$$p_X(x^v) = \sum_{j=1}^{K} \alpha_j p_{\mathcal{N}}(x^v; \mu_j, \Sigma_j), \qquad x^v \in \mathbb{R}^d, \tag{5.1}$$

with positive mixture components α_j that sum up to 1. The expression $p_{\mathcal{N}}(\,\cdot\,; \mu_j, \Sigma_j)$ denotes the Gaussian density function with mean $\mu_j \in \mathbb{R}^d$ and covariance matrices $\Sigma_j \in \mathbb{R}^{d\times d}$. In order to have a well-defined expression, we impose $\Sigma_j \succ 0$, i.e. we assume that the Σ_j are symmetric positive definite. Note that throughout this chapter, we suppose that the number of components K is given. A discussion of approaches for finding a suitable value for K is beyond the scope of this thesis, we refer to [95] for an overview of existing approaches.

In practice, the parameters of a Gaussian mixture model $\alpha_j, \mu_j, \Sigma_j$ are unknown and have to be estimated. Given iid observations $x_1, \dots, x_n$, a maximum likelihood approach is used, that is we seek to maximize the log-likelihood function

$$l(\tilde{\theta}) = \sum_{i=1}^{n} \log\left(p_X(x_i; \tilde{\theta})\right), \tag{5.2}$$

where $\tilde{\theta} = (\tilde{\theta}_1, \dots, \tilde{\theta}_K)$ with $\tilde{\theta}_j = (\alpha_j, \mu_j, \Sigma_j)$, see Definition 3.1. This yields the following constrained nonlinear optimization problem:

$$\max_{\substack{\alpha\in\Delta_K \\ \mu_j\in\mathbb{R}^d \\ \Sigma_j\succ 0}} \sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} \alpha_j p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j)\right), \tag{5.3}$$

where

$$\Delta_K = \{(\alpha_1, \dots, \alpha_K),\ \alpha_j \in (0,1) \forall j, \quad \sum_{j=1}^{K} \alpha_j = 1\} \tag{5.4}$$

is the probability simplex of dimension K [65]. We observe that the objective in (5.3) is not concave, for which reason the maximum likelihood estimator might not be unique and we have a multimodal distribution [101, Section 11.3.1]. In addition, finding the global optimum of (5.3) is an NP-hard problem [101, Section 11.3.1]. In this thesis, we focus on finding a local maximum of the problem (5.3).

The optimization problem (5.3) comes with constraints. We need to ensure that the mixing proportion variables α_j, $j = 1, \dots, K$ are between 0 and 1, and sum up to one. Further, we need to restrict the matrices Σ_j to be symmetric positive definite. Especially the latter is a challenge for classical nonlinear optimization algorithms as outlined in Section 4.1. Thus, in practice, the Expectation Maximization (EM) algorithm (Algorithm 7) is commonly used to iteratively solve the problem (5.3). For this, latent variables are introduced that model the membership to one of the K clusters with probability α_j, for details on the EM algorithm for Gaussian mixture models see Appendix B.1. For Gaussian mixture models, the EM algorithm turns out to consist of two simple steps, whereas the M-step (3.10) is concave and admits a closed-form solution (see Algorithm 8) [101, Section 11.4]. Though, as outlined in Section 3.2.2, the EM algorithm converges linearly if there is a high share of hidden information [94, Section 3.5]. For Gaussian mixture models, this means that many observations x_i cannot be clearly assigned to exactly one of the K Gaussians [95]. Considering the Gaussian mixture model as a probabilistic clustering model, a high share of hidden information yields the interpretation of a high overlap between clusters [147]. Motivated by EM's slow convergence, alternative approaches have been suggested such as using a hybrid approach of EM and nonlinear conjugate gradient or Newton iterations [4, 117] or using standard nonlinear optimization methods [147]. Here, the positive definiteness constraint is completely neglected [147] or a regularization term is added penalizing solutions close to the boundary [27]. Besides, search techniques like genetic and annealing algorithms [118, 110, 90] have recently been proposed for fitting Gaussian mixture models. The aforementioned methods suffer either from higher per-iteration costs or from missing convergence theory. For these reasons and thanks to its very low per-iteration costs the Expectation Maximization algorithm is the possibly most widely used algorithm for fitting Gaussian mixture models and is also the default method in familiar data mining software packages [108].

Recently, the work by Hosseini and Sra [65] has gained increasing interest for fitting Gaussian mixture models. The authors exploit the Riemannian geometry of the parameter space and formulate the problem as a Riemannian optimization problem. With this formulation, the authors of [65] propose a Riemannian nonlinear conjugate gradient method and a Riemannian LBFGS (limited-memory Broyden-Fletcher-Goldfarb-Shanno [105]) method by considering a product manifold involving the manifold of positive definite matrices. The authors point out that their method is particularly strong for highly overlapping clusters, where the share of hidden information is large and the EM algorithm shows linear convergence, only. In their follow-up paper [66], Hosseini and Sra further introduce a Riemannian stochastic gradient descent method to address problems with a larger number of observations. Furthermore, they propose a penalization term for problem (5.3) allowing for maximum a posterior estimation and avoiding numerical singularities. The works [65, 66] are the basis for this chapter. While the publications [65, 66] do not use exact second-order information, we introduce an explicit formula for the Riemannian Hessian of the GMM fitting problem which is used for the introduced Riemannian Newton trust-region method (R-NTR).

In the following, we first review the modification of the objective (5.3) introduced in [65, 66] from which Riemannian optimizers benefit in terms of convergence speed and numerical stability. We then derive the Riemannian gradient and the Riemannian Hessian, which is one of the main contributions of this thesis. Equipped with the Riemannian Hessian, we introduce the Riemannian Newton trust-region algorithm for fitting Gaussian mixture models in Section 5.3 and show numerical results of our method in Section 5.4.

5.2 Riemannian Approach for Gaussian Mixture Models

The tools of Riemannian optimization introduced in Chapter 2 and Chapter 4 allow us to apply geometric optimization techniques of the objective (5.3) by considering the constraint set as a product manifold. However, before directly applying Riemannian optimization on the objective (5.3), we introduce a modification of the objective as proposed by [65, 66].

5.2.1 Modification of the Objective

By setting $f = -l(\tilde{\theta})$, the problem (5.3) can be formulated as a Riemannian optimization problem of the form (2.9), see Chapter 4. Yet, Hosseini and Sra experimentally showed in their work [65] that such a formulation of the problem leads to Riemannian optimization methods showing inferior convergence compared to the EM algorithm. This can be mainly led back to the fact that the maximization in the M-step of EM, i.e. the maximization of the log-likelihood for a single Gaussian, is a concave problem and thus very easy to solve [65]. One of the reasons for EM's popularity for fitting Gaussian mixture models is that the M-step additionally admits a closed-form solution in the maximization step (see Algorithm 8, Appendix B.1). When considering the log-likelihood of a single Gaussian in a Riemannian setting, that is

$$\max_{\substack{\mu_j \in \mathbb{R}^d \\ \Sigma_j \succ 0}} \sum_{i=1}^{n} \log \left(p_{\mathcal{N}}(x_i; \mu, \Sigma) \right), \tag{5.5}$$

the objective is not geodesically concave in the sense of Definition 2.24 [65]. This means that the translation of $\sum_{i=1}^{n} \log \left(p_{\mathcal{N}}(\cdot \, ; \mu, \Sigma) \right)$ into a Riemannian setting adds a remarkable geometric mismatch [65]. A surrogate to this mismatch is to reformulate the objective in such a way such that the optimization problem with $K = 1$ is geodesically concave. Hosseini and Sra introduce an alternative equivalent formulation of the objective in [65] which corrects this geometric pathology. We explain the suggested reformulation in the following.

Addressing the geometric mismatch: reformulation of the objective

We augment the observed data $x_1, \ldots, x_n$ by concatenating them with the scalar 1, that is we introduce the data y_i, $i = 1, \ldots, n$ by setting

$$y_i = \begin{pmatrix} x_i \\ 1 \end{pmatrix} \in \mathbb{R}^{d+1}, \qquad i = 1, \ldots, n. \tag{5.6}$$

Further, we introduce the function $q_{\mathcal{N}}$ as the probability density function of a centered $d+1$-dimensional Gaussian random variable corrected by a constant factor, i.e.

$$q_{\mathcal{N}}(y_i; S_j) := \sqrt{2\pi} \exp \left(\frac{1}{2} \right) p_{\mathcal{N}}(y_i; 0, S_j). \tag{5.7}$$

Here, $p_{\mathcal{N}}(y_i; 0, S_j)$ denotes the Gaussian probability density function with $d+1$-dimensional mean 0 and covariance matrix $S_j \in \mathbb{R}^{(d+1)\times(d+1)}$ of a $d+1$-dimensional random variable Y evaluated at observation y_i. With this reformulation, we now mapped our objective function restricted to the estimation of the covariance matrices from the manifold $\mathbb{P}^d$ to the manifold $\mathbb{P}^{d+1}$. This yields the desired geodesic concavity of a single Gaussian component.

Theorem 5.1. *[65, Proposition 1] Define*

$$\Phi(S) = -\sum_{i=1}^{n} \log\left(q_{\mathcal{N}}(y_i; S)\right),$$

where $S \in \mathbb{P}^{d+1}$. Then, the function Φ is geodesically convex.

Proof. The proof is based on showing the inequality

$$\Phi\left(\gamma_{S_1,S_2}(t)\right) \leq \Phi(S_1) + \Phi(S_2),$$

where γ is the unique geodesic (4.9), see Definition 2.24. The proof is based on an inequality for the geodesic midpoint [17, Theorem 4.1.3] and can be found in [66]. $\square$

This means that by considering the centered Gaussian distribution of the augmented data (5.7), we have derived an analogue concavity property of the Riemannian problem compared to the (Euclidean) M-step in the EM algorithm: geodesic concavity of the log-likelihood of a Gaussian mixture model with $K = 1$ components. Considering the Gaussian mixture model with $K > 1$ components, we additionally need to ensure that the mixing proportions $\alpha = (\alpha_1, \ldots, \alpha_K)$ are in the probability simplex Δ_K, that is that each of the α_j's is strictly positive and that they sum up to one. This is achieved by introducing a variable $\eta \in \mathbb{R}^{K-1}$ and the constant $\eta_K = 0$ [65]. We define the α_j's as

$$\alpha_j := \frac{\exp(\eta_j)}{\sum\limits_{k=1}^{K} \exp(\eta_k)}, \qquad j = 1, \ldots, K, \tag{5.8}$$

such that the condition $\alpha = (\alpha_1, \ldots, \alpha_K) \in \Delta_K$ is automatically fulfilled by design. With these considerations in mind, we define the variable of interest as $\theta = (\theta_1, \ldots, \theta_K)$, where

$$\theta_j = (S_j, \eta_j) \quad j = 1, \ldots, K-1, \text{ and } \quad \theta_K = (S_K, 0).$$

The product manifold of interest is given by

$$\mathcal{M}_{GMM} = \bigtimes_{j=1}^{K} \left(\mathbb{P}^{d+1}\right) \times \mathbb{R}^{K-1}. \tag{5.9}$$

and the reformulated problem reads

$$\max_{\theta \in \mathcal{M}_{GMM}} \hat{\mathcal{L}}(\theta) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} h^i(\theta_j)\right), \tag{5.10}$$

where

$$h^i(\theta_j) = \frac{\exp(\eta_j)}{\sum_{k=1}^{K}\exp(\eta_j)} q_{\mathcal{N}}(y_i; S_j) = \frac{\exp(\eta_j)}{\sum_{k=1}^{K}\exp(\eta_k)} \frac{\exp\left(\frac{1}{2}\left(1 - y_i^T S_j^{-1} y_i\right)\right)}{\sqrt{(2\pi)^d \det(S_j)}}, \tag{5.11}$$

see (5.7). This is a Riemannian optimization problem where we corrected the geometric mismatch induced by translating the maximum-likelihood estimation task into a Riemannian framework. What remains to state is that such a reformulation is meaningful. Hosseini and Sra [65] show that the two problems (5.3) and (5.10) are equivalent in the following sense:

Theorem 5.2. *[65, Theorem 2.1 - 2.2] A local maximum of the reformulated GMM log-likelihood $\hat{\mathcal{L}}(\theta)$ as in (5.10) with maximizer $\theta^* = ({\theta_1}^*, \dots, {\theta_K}^*)$, ${\theta_j}^* = ({S_j}^*, {\eta_j}^*)$ is a local maximum of the original log-likelihood $l(\tilde{\theta}^*)$ defined in (5.2) with maximizer $\tilde{\theta}^* = ({\alpha_j}^*, {\mu_j}^*, {\Sigma_j}^*)_{j=1,\dots,K}$.*
Further, if $\tilde{\theta}^* = ({\alpha_j}^*, {\mu_j}^*, {\Sigma_j}^*)_j$ maximizes (5.3), then

$$\sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} {\alpha_j}^* p_{\mathcal{N}}(x_i; {\mu_j}^*, {\Sigma_j}^*)\right) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} h^i({\theta_j}^*)\right),$$

where $\theta_j^* = (S_j^*, \eta_j^*)$ and

$$\begin{aligned} S_j^* &= \begin{pmatrix} \Sigma_j^* + \mu_j^* \mu_j^{*T} & \mu_j^* \\ \mu_j^{*T} & 1 \end{pmatrix}, \qquad (5.12) \\ \eta_j^* &= \log\left(\frac{\alpha_j^*}{\alpha_K^*}\right) \quad j = 1, \dots, K-1; \quad \eta_K \equiv 0 \end{aligned}$$

Proof. The proof is based on decomposing the matrices S_j via Schur complements and can be found in [65] and [66]. □

This means that there is a one-to-one mapping between a local maximizer of the original problem (5.3) and the reformulated problem (5.10) and the two objectives coincide in a local optimum. Hosseini and Sra experimentally showed in their works [65, 66] that such a reformulation brings the desired faster convergence of their proposed Riemannian optimization algorithms as expected. For this reason, we consider the reformulated problem (5.10) in the following.

Evading singularity: regularization of the objective

A problem with the objective $\hat{\mathcal{L}}$ in (5.10) is its unboundedness from above which is also observed and well studied for the original problem in (5.3), e.g. by [106, 125].

Theorem 5.3. *[124, Theorem 7] The reformulated objective* (5.10) *is unbounded from above.*

Proof. We omit the proof as it is a straightforward extension to the proof for unboundedness of the original problem; the proof can be found in [125]. □

The unboundedness of the objective (5.10) makes convergence theory hard to study as it might occur that a local maximum of the Riemannian optimization problem does not even exist [105]. This problem is well-known also in the Euclidean setting, and has been studied extensively in the literature [38, 59, 106] in particular for the EM algorithm for Gaussian mixture models. For the EM algorithm, a typical remedy to this consists in considering a *penalized maximum likelihood* approach: For the original problem in (5.3), a penalization based on a Bayesian setting is used. The works [106, 125] suggest to modify the objective (5.3) by adding a specific regularization term. Such a regularization is equivalent to considering the maximum a posteriori likelihood estimator (see Definition 3.1, 4.). The authors of [106, 125] suggest to use conjugate priors for the Bayesian setting in order to keep the structure of the estimates. These come with the advantage that the posterior density function $g_{\theta|X}(\theta|x)$ is in the same probability distribution family as the prior density function $g_\theta(\theta)$ (see Appendix A.2). This penalization yields the objective

$$\begin{aligned} l_{\text{pen}}(\tilde{\theta}) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} \alpha_j p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j)\right) + \log\left(g_\alpha^{dirich}(\alpha; \zeta)\right) \\ + \sum_{j=1}^{K} \left(g_{\mu_j}^{\mathcal{N}}(\mu_j; \lambda, \kappa, \Sigma) + \log g_{\Sigma_j}^{Wi}(\Sigma_j^{-1}|\Lambda, \nu)\right), \end{aligned} \tag{5.13}$$

where g_α^{dirich} denotes the conjugate prior of α, the density g_{μ_j} denotes the conjugate prior of μ_j and $g_{\Sigma_j}(\Sigma_j^{-1}|\Lambda, \nu)$ the conjugate prior of Σ_j. The formulas and details for these expressions are given in Appendix B.2. For the penalized log-likelihood (5.13), we can derive a penalized EM algorithm which comes with the same simplicity as the original EM algorithm for Gaussian mixture models (see Appendix B.2, Algorithm 9).

An according penalization term has been suggested by Hosseini in Sra in their recent work [66] for the reformulated objective. The cruciality here is that the geodesic concavity for the problem with $K = 1$ is not lost by the regularizer, and that the relationship between a local maximizer of the original problem and a local maximizer of the reformulated problem (5.12) still holds [66]. The following penalizer $\mathrm{Pen}(\theta; \Psi, \zeta)$ meets these requirements:

$$\mathrm{Pen}(\theta; \Psi, \zeta) = \sum_{j=1}^{K} \psi(S_j, \Psi) + \varphi(\eta, \zeta). \tag{5.14}$$

Here, the penalizer $\psi(S_j, \Psi)$ for the covariance matrices S_j reads

$$\psi(S_j, \Psi) = -\frac{\rho}{2} \log\det(S_j) - \frac{\beta}{2} \operatorname{tr}(\Psi {S_j}^{-1}), \tag{5.15}$$

where $\rho = \gamma(d + \nu + 1) + \beta$, $\gamma, \beta > 0$, and Ψ is a block matrix of the form

$$\Psi = \begin{pmatrix} \frac{\gamma}{\beta}\Lambda + \kappa\lambda\lambda^T & \kappa\lambda \\ \kappa\lambda^T & \kappa \end{pmatrix}$$

for fixed parameters $\Lambda \in \mathbb{R}^{d\times d}$, $\Lambda \succ 0$, $\lambda \in \mathbb{R}^d$ and $\kappa > 0$. The penalizer for the mixing proportions is given by

$$\varphi(\eta, \zeta) = \zeta \left(\sum_{j=1}^{K} \eta_j - K \log \left(\sum_{k=1}^{K} \exp(\eta_k) \right) \right), \tag{5.16}$$

where $\zeta > 0$. It is important to note that the parameters $\lambda, \kappa, \Lambda, \nu$ are the same as in the additive penalizer (5.13). This allows to state the following important theorem for the reformulated penalized log-likelihood function.

Theorem 5.4. *[66, Corollary 6] Consider the penalized reformulated log-likelihood*

$$\hat{\mathcal{L}}_{\mathrm{pen}}(\theta; \Psi, \zeta) = \hat{\mathcal{L}}(\theta) + \mathrm{Pen}(\theta; \Psi, \zeta) \tag{5.17}$$

with $\hat{\mathcal{L}}$ *as in* (5.10) *and* Pen *as in* (5.14).
Assume that

$$\gamma = \beta \frac{\kappa - 1}{d + \nu + 1}$$

for the parameter γ *in* $\rho = \gamma(d + \nu + 1) + \beta$ *occuring in* (5.15). *Then, the objective* (5.17) *is geodesically concave for a single component, that is* $K = 1$. *Further, a local maximum of the reformulated penalized GMM log-likelihood* $\hat{\mathcal{L}}_{\mathrm{pen}}(\theta)$ *with maximizer* $\theta^* = ({\theta_1}^*, \dots, {\theta_K}^*)$, ${\theta_j}^* = ({S_j}^*, {\eta_j}^*)$ *is a local maximum of the original penalized log-likelihood* $l_{\mathrm{pen}}(\tilde{\theta}^*)$ *defined in* (5.13) *with maximizer* $\tilde{\theta}^* = ({\alpha_j}^*, {\mu_j}^*, {\Sigma_j}^*)_{j=1,\dots,K}$. *If* $\tilde{\theta}^* = ({\alpha_j}^*, {\mu_j}^*, {\Sigma_j}^*)_j$ *maximizes* (5.13), *then the maximizers* $\tilde{\theta}^*$ *and* θ^* *are related via* (5.12) *in Theorem 5.2.*

Proof. A proof can be found in [66]. □

Theorem 5.4 ensures that we keep the beneficial structure of the introduced reformulation when adding the regularization term $\mathrm{Pen}(\theta; \Psi, \zeta)$.

We recall that the initial motivation for using maximum a posterior estimation was that the reformulated problem (5.10) is unbounded from above. We now show that the penalized reformulated problem (5.17) is bounded from above.

Theorem 5.5. *The penalized optimization problem in* (5.17) *is bounded from above.*

Proof. We follow the proof for the original objective (5.13) as in [125]. The penalized objective reads

$$\hat{\mathcal{L}}_{\mathrm{pen}}(\theta; \Psi, \zeta) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} q_{\mathcal{N}}^{\mathrm{AP}}(\theta_j, S_j; \Psi, \zeta) \right),$$

where

$$q_{\mathcal{N}}^{\mathrm{AP}}(\theta_j, S_j; \Psi, \zeta) = h^i(\theta_j) \Bigg(\prod_{k=1}^{K} \det(S_k)^{-\frac{\varrho}{2}} \exp\left(-\frac{1}{2}\operatorname{tr}(S_k^{-1}\Psi)\right) \alpha_k{}^{\zeta} \Bigg)^{1/n}.$$

We get the upper bound

$$\begin{aligned}
q_{\mathcal{N}}^{\mathrm{AP}}(\theta_j, S_j;\Psi, \zeta) &\leq h^i(\theta_j) \prod_{k=1}^{K} \det(S_k)^{-\frac{\varrho}{2}} \exp\left(-\frac{1}{2}\operatorname{tr}(S_k^{-1}\Psi)\right) \alpha_k^{\zeta} \\
&\leq a\alpha_j \det(S_j)^{-\frac{d}{2}} \prod_{k=1}^{K} \det(S_k)^{-\frac{\varrho}{2}} \exp\left(-\frac{1}{2}\operatorname{tr}(S_k^{-1}\Psi)\right) \\
&= a\alpha_j (\det(S_j))^{-\frac{d+\varrho}{2}} \exp\left(-\frac{1}{2}\operatorname{tr}(S_j^{-1}\Psi)\right) \prod_{\substack{k=1 \\ k\neq j}}^{K} \det(S_k)^{-\frac{\varrho}{2}} \exp\left(-\frac{1}{2}\operatorname{tr}(S_k^{-1}\Psi)\right),
\end{aligned} \tag{5.18}$$

where we applied Bernoulli's inequality in the first inequality and used the positive definiteness of S_j in the second inequality. The constant a is positive and independent of S_j and S_k.

By applying the relationship $\det(A)^{1/d} \leq \frac{1}{d}\operatorname{tr}(A)$ for a matrix $A \in \mathbb{R}^{d\times d}$ by the inequality of arithmetic and geometric means, we get for the right hand side of (5.18) the inequality

$$\det(S_k)^{-\frac{b}{2}} \exp(-\frac{1}{2}\operatorname{tr}(S_k^{-1}\Psi)) \leq (\det(S_k))^{-\frac{b}{2}} \exp\left(-\frac{d+1}{2}\left(\frac{\det(\Psi)}{\det(S_k)}\right)^{\frac{1}{d+1}}\right) \tag{5.19}$$

for some constant $b > 0$. The crucial part on the right side of (5.19) is when one of the S_k approaches a singular matrix and thus the determinant approaches zero. This is the situation where we reach the boundary of the parameter space $\mathbb{P}^{d+1}$. We study this issue in further detail: Without loss of generality , let $k = 1$ be the component where we approach the boundary. Formally, let S_1^* be a singular positive semidefinite matrix of rank $r < d+1$. Then, there exists a decomposition of the form

$$S_1^* = U^T D U,$$

where $D = \operatorname{diag}(0, \ldots, 0, \lambda_{d-r}, \lambda_{d-r+1}, \ldots, \lambda_{d+1})$, $\lambda_l > 0$ for $l = d-r, \ldots, d+1$ and U an orthogonal square matrix of size $d+1$. Now consider the sequence $S_1^{(m)}$ given by

$$S_1^{(m)} = U^T D^{(m)} U, \tag{5.20}$$

where

$$D^{(m)} = \operatorname{diag}(\lambda_1^{(m)}, \ldots, \lambda_{d-r-1}^{(m)}, \lambda_{d-r}, \lambda_{d-r+1}, \ldots, \lambda_{d+1})$$

with $\left(\lambda_l^{(m)}\right)_{l=1,\ldots,d-r-1}$ converging to 0 as $m \to \infty$.

Then, the matrix $S_1^{(m)}$ converges to $S_1^{(*)}$. Setting $\lambda^{(m)} = \prod_1^{d-r-1} \lambda_l^{(m)}$ and $\lambda^+ = \prod_{d-r}^{d+1} \lambda_l$, the right hand side of (5.19) reads

$$\left(\lambda^{(m)}\lambda^+\right)^{-\frac{b}{2}} \exp\left(-\frac{d+1}{2}\left(\frac{\det(\Psi)}{\lambda^+\lambda^{(m)}}\right)^{\frac{1}{d+1}}\right),$$

which converges to 0 as $m \to \infty$ by the rule of L'Hôpital. This yields the required boundedness from above of the objective $\hat{\mathcal{L}}_{\text{pen}}$ at the boundary of the parameter space.
□

We now have derived a bounded objective where we corrected the geometric mismatch. This reformulation of the objective is used for Riemannian optimization. In the next section, we formulate the Riemannian optimization problem and derive formulas for the Riemannian gradient and the Riemannian Hessian.

5.2.2 Riemannian Setting, Gradient and Hessian

In the previous section, we have discussed some challenges to perform Riemannian optimization on the standard maximum likelihood formulation for Gaussian mixture models. We introduced a beneficial reformulation and a regularization term as suggested by Hosseini and Sra in [65, 66]. In the following, we specify the Riemannian setting for our modified optimization problem.

Problem:

Let $\theta = (\theta_1, \dots, \theta_K)$ with $\theta_j = (S_j, \eta_j)$, $S_j \succ 0$, $S_j \in \mathbb{R}^{d+1 \times d+1}$ and $\eta_j \in \mathbb{R}$. For given $x_1, \dots, x_n \in \mathbb{R}^d$, $\Psi \in \mathbb{R}^{d+1 \times d+1}$ and $\zeta > 0$, we consider the Riemannian optimization problem

$$\max_{\theta \in \mathcal{M}_{GMM}} \hat{\mathcal{L}}_{\text{pen}}(\theta) = \hat{\mathcal{L}}(\theta) + \text{Pen}(\theta; \Psi, \zeta), \tag{5.21}$$

where

$$\hat{\mathcal{L}}(\theta) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} h^i(\theta_j) \right), \quad h^i(\theta_j) = \frac{\exp(\eta_j)}{\sum\limits_{k=1}^{K} \exp(\eta_k)} \frac{\exp\left(\frac{1}{2}\left(1 - y_i^T S_j^{-1} y_i\right)\right)}{\sqrt{(2\pi)^d \det(S_j)}}$$

and

$$\text{Pen}(\theta; \Psi, \zeta) = -\frac{1}{2} \sum_{j=1}^{K} \rho \log \det(S_j) + \beta \operatorname{tr}(\Psi S_j^{-1}) + \zeta \left(\sum_{j=1}^{K} \eta_j - K \log \left(\sum_{k=1}^{K} \exp(\eta_k) \right) \right).$$

Further,

$$\mathcal{M}_{GMM} = \bigtimes_{j=1}^{K} \left(\mathbb{P}^{d+1} \right) \times \mathbb{R}^{K-1}. \tag{5.22}$$

Here, the observations $y_1, \dots, y_n$ are related to $x_1, \dots, x_n$ by (5.6), that is

$$y_i = \begin{pmatrix} x_i \\ 1 \end{pmatrix} \in \mathbb{R}^{d+1}, \qquad i = 1, \dots, n.$$

Riemannian setting for Gaussian mixture models

The manifold of interest (5.22) is a product manifold which consists of K times the manifold of positive definite matrices of dimension $d+1$, denoted by $\mathbb{P}^{d+1}$, and the natural Euclidean manifold $\mathbb{R}^{K-1}$. With the specification of the manifold $\mathbb{P}^{d+1}$ discussed in Chapter 4, we state important Riemannian characteristics for our optimization problem (5.21).

The tangent space of $\mathbb{P}^{d+1}$ is given by the set of symmetric matrices of dimension $d+1$, see Section 4.2. Thus, the corresponding tangent space for $\mathcal{M}_{GMM}$ reads

$$T_\theta \mathcal{M}_{GMM} = \left(\mathbb{S}^{d+1} \right)^K \times \mathbb{R}^{K-1}, \tag{5.23}$$

where $\mathbb{S}^{d+1}$ is given by (4.2).

A metric on this tangent space $T_\theta\mathcal{M}_{GMM}$ can be constructed by metrics defined on the component-wise tangent spaces $\mathbb{S}^{d+1}$ and $\mathbb{R}^{K-1}$, see (2.2). Following the discussion about Riemannian metrics for covariance matrices in Section 4.2, we choose the affine-invariant metric (4.6) for the manifold $\mathbb{P}^{d+1}$. For the Euclidean manifold $\mathbb{R}^{K-1}$, we choose the standard Euclidean inner product. Let $\theta \in \mathcal{M}_{GMM}$ and $\xi_\theta, \chi_\theta \in T_\theta\mathcal{M}_{GMM}$, that is

$$\xi_\theta = (\xi_{\theta_1}, \dots, \xi_{\theta_K}), \qquad \xi_{\theta_j} = (\xi_{S_j}, \xi_{\eta_j}) \tag{5.24}$$

$$\chi_\theta = (\chi_{\theta_1}, \dots, \chi_{\theta_K}), \qquad \chi_{\theta_j} = (\chi_{S_j}, \chi_{\eta_j}). \tag{5.25}$$

Then, we define an inner product $\langle\cdot,\cdot\rangle_\theta$ on $T_\theta\mathcal{M}_{GMM}$ as

$$\langle \xi_\theta, \chi_\theta\rangle_\theta = \sum_{j=1}^{K} \operatorname{tr}(S_j^{-1}\xi_{S_j}S_j^{-1}\chi_{S_j}) + \xi_\eta^T\chi_{\eta\cdot}, \tag{5.26}$$

according to Section 2.1.3.

A retraction $R_\theta(\xi_\theta)$ mapping points from the tangent space $T_\theta\mathcal{M}_{GMM}$ back onto the manifold $\mathcal{M}_{GMM}$ is given by

$$R_\theta(\xi_\theta) = \begin{pmatrix} \left(S_j \exp\left(S_j^{-1}\xi_{S_j}\right)\right)_{j=1,\dots,K} \\ (\eta_j + \xi_{\eta_j})_{j=1,\dots,K-1} \end{pmatrix}, \tag{5.27}$$

where the first component is the exponential map with respect to the affine-invariant metric for positive definite matrices, see (4.16), and the second component is the natural exponential map on the Euclidean manifold.

The Riemannian gradient for Gaussian mixture models

We specify the Riemannian gradient for the problem (5.21) in the following based on the inner product (5.26). For better readability, we fix the regularization parameters Ψ, ζ in the penalizer $\operatorname{Pen}(\theta;\Psi,\zeta)$ in (5.14) and simply write $\operatorname{Pen}(\theta)$, $\hat{\mathcal{L}}_{\text{pen}}(\theta)$ for $\operatorname{Pen}(\theta;\Psi,\zeta)$, $\hat{\mathcal{L}}_{\text{pen}}(\theta;\Psi,\zeta)$ in the following.

Theorem 5.6. *For $\theta \in \mathcal{M}_{GMM}$, the Riemannian gradient $\operatorname{grad}\hat{\mathcal{L}}_{\text{pen}}(\theta) \in T_\theta\mathcal{M}_{GMM}$ of problem (5.21) is given by*

$$\operatorname{grad}\hat{\mathcal{L}}_{\text{pen}}(\theta) = \begin{pmatrix} \left(\frac{1}{2}\left(\sum\limits_{i=1}^{n} f_l^i(y_i{y_i}^T - S_l) - (\rho S_l - \beta\Psi)\right)\right)_{l=1,\dots,K} \\ \left(\sum\limits_{i=1}^{n} f_r^i - \alpha_r \sum\limits_{j=1}^{K} f_j^i + \zeta\left(1 - K\alpha_r\right)\right)_{r=1,\dots,K-1} \end{pmatrix}, \tag{5.28}$$

where

$$f_l^i = \frac{h^i(\theta_l)}{\sum\limits_{j=1}^K h^i(\theta_j)}, \qquad \alpha_r = \frac{\exp(\eta_r)}{\sum\limits_{k=1}^K \exp(\eta_k)}.$$

Proof. We find the Riemannian gradient of a function defined on a product manifold by the component-wise gradient, that is the gradient can be expressed as

$$\operatorname{grad} \hat{\mathcal{L}}_{\text{pen}}(\theta) = \begin{pmatrix} \left(\operatorname{grad}_{S_l} \hat{\mathcal{L}}_{\text{pen}}(\theta)\right)_{l=1,\dots,K} \\ \left(\operatorname{grad}_{\eta_r} \hat{\mathcal{L}}_{\text{pen}}(\theta)\right)_{r=1,\dots,K-1} \end{pmatrix}$$

where $\operatorname{grad}_{S_l} \hat{\mathcal{L}}_{\text{pen}}(\theta) \in \mathbb{S}^{d+1}$, $\operatorname{grad}_{\eta_r} \hat{\mathcal{L}}_{\text{pen}}(\theta) \in \mathbb{R}$ denotes the gradient with respect to S_l η_r (at position S_l, η_r), respectively, that is we fix all variables different from S_l, η_r and differentiate with respect to S_l, η_r, respectively.
We start with the variable $\eta \in \mathbb{R}^{K-1}$. Its gradient is the classical Euclidean gradient, hence we get by using the chain rule

$$\operatorname{grad}_{\eta_r} \hat{\mathcal{L}}(\theta) = \sum_{i=1}^n \left(\sum_{k=1}^K h^i(\theta_k)\right)^{-1} \sum_{j=1}^K \frac{\partial h^i(\theta_j)}{\partial \eta_l} = \sum_{i=1}^n \sum_{j=1}^K f_j^i \left(\mathbb{1}_{\{j=r\}} - \frac{\exp(\eta_r)}{\sum\limits_{k=1}^K \exp(\eta_k)} \right)$$

for $r = 1, \dots, K-1$, where $\mathbb{1}_{\{j=r\}} = 1$ if $j = r$ and 0, else.
For the derivative of the penalizer with respect to η_r, we get

$$\operatorname{grad}_{\eta_r} \operatorname{Pen}(\theta) = \zeta \left(1 - K \frac{\exp(\eta_r)}{\sum\limits_{j=1}^K \exp(\eta_j)} \right).$$

The Riemannian gradient with respect to S_j under the described setting is given by the projected Euclidean gradient onto the tangent space $\mathbb{S}^{d+1}$, that is, see (4.12),

$$\operatorname{grad}_{S_l} \hat{\mathcal{L}}_{\text{pen}}(\theta) = \frac{1}{2} S_l \left(\operatorname{grad}^e_{S_l} \hat{\mathcal{L}}_{\text{pen}}(\theta) + \left(\operatorname{grad}^e_{S_l} \hat{\mathcal{L}}_{\text{pen}}(\theta)\right)^T \right) S_l. \tag{5.29}$$

Here, $\operatorname{grad}^e_{S_l} \hat{\mathcal{L}}_{\text{pen}}(\theta)$ denotes the Euclidean gradient of the Euclidean smooth extension of $\hat{\mathcal{L}}_{\text{pen}}(\theta)$ with respect to S_l. In a first step, we thus compute the Euclidean gradient with respect to a matrix S_l. We obtain

$$\operatorname{grad}^e_{S_l} \hat{\mathcal{L}}(\theta) = -\frac{1}{2} \sum_{i=1}^n f_l^i (S_l^{-1} y_i {y_i}^T S_l^{-1} - S_l^{-1}), \tag{5.30}$$

where we used the Leibniz rule and the partial matrix derivatives

$$\begin{aligned}
\frac{\partial\left(\det(S_l)^{-1/2}\right)}{\partial S_l} &= -\frac{1}{2}(\det(S_l))^{-1/2} S_l^{-1}, \\
\frac{\partial \exp\left(-\frac{1}{2} y_i^T S_l^{-1} y_i\right)}{\partial S_l} &= \frac{1}{2} \exp\left(-\frac{1}{2} y_i^T S_l^{-1} y_i\right) S_l^{-1} y_i {y_i}^T S_l^{-1}
\end{aligned}$$

which holds by the chain rule and the fact that S_l^{-1} is symmetric. Using the relationship (5.29) and using (5.30) yields the Riemannian gradient with respect to S_l. It is given by

$$\operatorname{grad}_{S_l} \hat{\mathcal{L}}(\theta) = \frac{1}{2} \sum_{i=1}^{n} f_l^i (y_i {y_i}^T - S_l).$$

Analogously, we compute the Euclidean gradient of the matrix penalizer $\psi(S_j, \Phi)$ and use the relationship (5.29) to get the Riemannian gradient of the matrix penalizer (5.15). □

A nonlinear Riemannian conjugate gradient and a Riemannian LBFGS method for Gaussian mixture models without penalization term have been proposed in the work [65] by Hosseini and Sra. The authors experimentally showed that these methods could compete with the Expectation Maximization algorithm. In their follow-up work [66], the authors introduced a Riemannian stochastic gradient descent method based on the penalized reformulated problem (5.21), allowing for improved scalability if the number of observations n increases. However, the methods suggested by the authors do not use exact second-order information via the Riemannian Hessian. Besides, using methods like the nonlinear conjugate gradient method or the LBFGS method in a Riemannian setting requires vector transport (2.25) which potentially goes along with additional numerical costs. A contribution of this thesis is the establishment of an explicit formula for the Hessian which allows for Newton's method. Newton-type methods are in general expected to show better convergence rates than the investigated methods, see [2, 105]. We derive a formula for the Riemannian Hessian in the following.

Riemannian Hessian for Gaussian Mixture Models

The following theorem states a formula for the Hessian of the penalized reformulated problem (5.21).

Theorem 5.7. *Let $\theta \in \mathcal{M}_{GMM}$ and $\xi_\theta \in T_\theta \mathcal{M}_{GMM}$. The Riemannian Hessian of $\hat{\mathcal{L}}_{\text{pen}}$ along the direction ξ_θ is given by*

$$\operatorname{Hess} \hat{\mathcal{L}}_{\text{pen}}(\theta)[\xi_\theta] = (\zeta_S, \zeta_\eta) \in T_\theta \mathcal{M}_{GMM},$$

where $\zeta_S = (\zeta_{S_1}, \dots, \zeta_{S_k})$, $\zeta_\eta = (\zeta_{\eta_1}, \dots, \zeta_{\eta_{K-1}})$ *and*

$$\zeta_{S_l} = -\frac{1}{4}\left(\sum_{i=1}^{n} f_l{}^i \left[C_l{}^i - \left(a_l{}^i - \sum_{j=1}^{K} f_j{}^i a_j{}^i\right)(y_i y_i{}^T - S_l)\right] - \beta\left(\Psi S_l{}^{-1}\xi_{S_l} + \xi_{S_l} S_l{}^{-1}\Psi\right)\right), \tag{5.31}$$

$$\zeta_{\eta_r} = \frac{1}{2}\sum_{i=1}^{n}\left[f_r{}^i\left(a_r{}^i - \sum_{j=1}^{K} f_j{}^i a_j{}^i\right) - 2\alpha_r\left(\xi_{\eta_r} - \sum_{j=1}^{K-1}\alpha_j\xi_{\eta_j}\right)\right] + K\zeta\alpha_r\left(\xi_{\eta_r} - \sum_{j=1}^{K-1}\alpha_j\xi_{\eta_j}\right) \tag{5.32}$$

for $l = 1, \dots, K$, $r = 1, \dots, K-1$ *and*

$$a_l{}^i = y_i{}^T S_l{}^{-1}\xi_{S_l} S_l{}^{-1} y_i - \operatorname{tr}(S_l^{-1}\xi_{S_l}) + 2\xi_{\eta_l}, \quad f_l{}^i = \frac{h^i(\theta_l)}{\sum_{j=1}^{K} h^i(\theta_j)}, \qquad \alpha_r = \frac{\exp(\eta_r)}{\sum_{k=1}^{K}\exp(\eta_k)},$$

$$C_l{}^i = y_i y_i{}^T S_l{}^{-1}\xi_{S_l} + \xi_{S_l} S_l{}^{-1} y_i y_i{}^T, \qquad \xi_{\eta_K} \equiv 0.$$

Proof. From the relationship (2.5), we get

$$\begin{aligned}\operatorname{Hess}\hat{\mathcal{L}}_{\text{pen}}(\theta)[\xi_\theta] &= \nabla_\theta \operatorname{grad}\hat{\mathcal{L}}_{\text{pen}}(\theta)\\ &= \left(\left(\nabla^{pd}_{\xi_{S_l}}\operatorname{grad}\hat{\mathcal{L}}_{\text{pen}}(\theta)\right)_{l=1,\dots,K}, \left(\nabla^{e}_{\xi_{\eta_r}}\operatorname{grad}\hat{\mathcal{L}}_{\text{pen}}(\theta)\right)_{r=1,\dots,K-1}\right)^T,\end{aligned} \tag{5.33}$$

where $\nabla^{pd}_{\xi_{S_l}}$ denotes the Riemannian connection for positive definite matrices specified in (4.13) and $\nabla^{e}_{\xi_{\eta_r}}$ denotes the classical Euclidean vector field differentiation along direction ξ_{η_r}. We observe that

$$\begin{aligned}&\operatorname{Hess}\hat{\mathcal{L}}_{\text{pen}}(\theta)[\xi_\theta]\\ &= \left(\left(\nabla^{pd}_{\xi_{S_l}}\operatorname{grad}\hat{\mathcal{L}}(\theta) + \nabla^{pd}_{\xi_{S_l}}\operatorname{grad}\operatorname{Pen}(\theta)\right)_l, \left(\nabla^{e}_{\xi_{\eta_r}}\operatorname{grad}\hat{\mathcal{L}}(\theta) + \nabla^{e}_{\xi_{\eta_r}}\operatorname{grad}\operatorname{Pen}(\theta)\right)_r\right)^T.\end{aligned} \tag{5.34}$$

We will now specify the single components of (5.34). In the following, we derive the expressions $\nabla^{(pd)}_{\xi_{S_l}}\operatorname{grad}\hat{\mathcal{L}}(\theta)$ and $\nabla^{e}_{\xi_{\eta_r}}\operatorname{grad}\hat{\mathcal{L}}(\theta)$ in detail, the derivation of $\nabla^{(pd)}_{\xi_{S_l}}\operatorname{grad}\operatorname{Pen}(\theta)$ and $\xi^{e}_{\eta_r}\operatorname{grad}\operatorname{Pen}(\theta)$ is straightforward.

We denote the gradient $\operatorname{grad}\hat{\mathcal{L}}(\theta)$ at position (with respect to) S_l, η_r by $\operatorname{grad}_{S_l}\hat{\mathcal{L}}_{\text{pen}}(\theta)$, $\operatorname{grad}_{\eta_r}\hat{\mathcal{L}}_{\text{pen}}(\theta)$, respectively, as in the proof of Theorem 5.6.

For the Hessian of $\hat{\mathcal{L}}$ at position η_r, we observe that the Riemannian connection $\nabla^e_{\xi_{\eta_r}}$ for $\xi_{\eta_r} \in \mathbb{R}$ is the classical vector field differentiation. We obtain

$$\nabla^e_{\xi_{\eta_r}} \operatorname{grad} \hat{\mathcal{L}}(\theta) = \sum_{j=1}^{K} \mathrm{D}_{S_j}(\operatorname{grad}_{\eta_r} \hat{\mathcal{L}}(\theta))[\xi_{S_j}] + \sum_{j=1}^{K-1} \mathrm{D}_{\eta_r}(\operatorname{grad}_{\eta_j} \hat{\mathcal{L}}(\theta))[\xi_{\eta_j}], \tag{5.35}$$

where $\mathrm{D}_{S_j}(\,\cdot\,)[\xi_{S_j}]$, $\mathrm{D}_{\eta_r}(\,\cdot\,)[\xi_{\eta_r}]$ denote the classical directional derivatives with respect to S_j, η_j along the directions ξ_{S_j} and ξ_{η_j}, respectively.
For the first part on the right hand side of (5.35), we have

$$\begin{aligned}\sum_{j=1}^{K} \mathrm{D}_{S_j}(\operatorname{grad}_{\eta_r} \hat{\mathcal{L}}(\theta))[\xi_{S_j}] = \frac{1}{2}\sum_{i=1}^{n} \Bigg[\frac{h^i(\theta_r)}{\sum\limits_{k=1}^{K} h^i(\theta_k)} \Big(& {y_i}^T {S_r}^{-1} \xi_{S_r} {S_r}^{-1} y_i - \operatorname{tr}(S_r^{-1}\xi_{S_r}) \\ & - \sum_{j=1}^{K} \frac{h^i(\theta_j)}{\sum\limits_{k=1}^{K} h^i(\theta_j)} ({y_i}^T {S_j}^{-1} \xi_{S_j} {S_j}^{-1} y_i - \operatorname{tr}(S_j^{-1}\xi_{S_j})) \Big) \Bigg]\end{aligned}$$

and for the second part

$$\begin{aligned}\sum_{j=1}^{K-1} \mathrm{D}_{\eta_j}(\operatorname{grad}_{\eta_r} \hat{\mathcal{L}}(\theta))[\xi_{\eta_j}] = \sum_{i=1}^{n} \Bigg[\Bigg(& \frac{h^i(\theta_r)}{\sum\limits_{k=1}^{K} h^i(\theta_k)} - \alpha_r \Bigg) \xi_{\eta_r} + \alpha_r \sum_{j=1}^{K-1} \alpha_j \xi_{\eta_j} \\ & - \frac{h^i(\theta_r)}{\sum\limits_{k=1}^{K} h^i(\theta_j)} \sum_{j=1}^{K-1} \frac{h^i(\theta_j)}{\sum\limits_{k=1}^{K} h^i(\theta_j)} \xi_{\eta_j} \Bigg]\end{aligned}$$

by applying the chain rule, the Leibniz rule and the relationship $\alpha_r = \frac{\exp(\eta_r)}{\sum_{k=1}^{K} \exp(\eta_k)}$.
Plugging the terms into (5.35), this yields the expression for ζ_{η_r} in (5.32) together with a straightforward derivation for the penalization term.

For the Hessian with respect to the matrices S_l, we use the Riemannian connection ∇^{pd} for the manifold of positive definite matrices with the affine-invariant metric as specified in (4.13). Hence, for the first part in (5.34), we get

$$\begin{aligned}\Big(\nabla^{(pd)}_{\xi_{S_l}} \operatorname{grad} \hat{\mathcal{L}}(\theta)\Big)_l = \Bigg(& \sum_{j=1}^{K} \mathrm{D}_{S_j}(\operatorname{grad}_{S_l} \hat{\mathcal{L}}(\theta))[\xi_{S_j}] + \sum_{j=1}^{K-1} \mathrm{D}_{\eta_j}(\operatorname{grad}_{S_l} \hat{\mathcal{L}}(\theta))[\xi_{\eta_j}] \\ & - \frac{1}{2}\Big(\operatorname{grad}_{S_l} \hat{\mathcal{L}}(\theta) S_l^{-1} \xi_{S_l} + \xi_{S_l} S_l^{-1} \operatorname{grad}_{S_l} \hat{\mathcal{L}}(\theta)\Big)\Bigg)_l .\end{aligned} \tag{5.36}$$

After applying the chain rule and the Leibniz rule, we obtain

$$\sum_{j=1}^{K} \mathrm{D}_{S_j}(\mathrm{grad}_{S_l}\, \hat{\mathcal{L}}(\theta))[\xi_{S_j}] = -\frac{1}{4}\sum_{i=1}^{n} f_l^i \Bigg[2\xi_{S_l} - \bigg(({y_i}^T {S_l}^{-1} \xi_{S_l} {S_l}^{-1} y_i - \mathrm{tr}({S_l}^{-1}\xi_{S_l})) + \sum_{j=1}^{K} f_j^i ({y_i}^T {S_j}^{-1}\xi_{S_j}{S_j}^{-1} y_i - \mathrm{tr}({S_j}^{-1}\xi_{S_j})) \bigg) (y_i {y_i}^T - S_l) \Bigg] \tag{5.37}$$

and

$$\sum_{j=1}^{K-1} \mathrm{D}_{\eta_j}(\mathrm{grad}_{S_l}\, \hat{\mathcal{L}}(\theta))[\xi_{\eta_j}] = \frac{1}{2}\sum_{i=1}^{n} \left(\frac{h^i(\theta_l)}{\sum\limits_{k=1}^{K} h^i(\theta_j)} \left(\xi_{\eta_l} - \sum_{j=1}^{K-1} \frac{h^i(\theta_j)}{\sum\limits_{k=1}^{K} h^i(\theta_k)} \xi_{\eta_k} \right) \right) (y_i {y_i}^T - S_l). \tag{5.38}$$

We plug (5.37), (5.38) into (5.36) and use the Riemannian gradient (5.28) at position S_l for the last term in (5.36). After some rearrangement of terms and an analogous derivation for the additive penalizer, we obtain the expression for ζ_{S_l} in (5.32). □

With the availabity of an explicit formula for the Riemannian Hessian, we can build richer Newton-type algorithms for Gaussian mixture models. Considering the formulas derived for the gradient and the Hessian, we observe that when an observation x_i has low probability of being drawn from a Gaussian component l, that is a low value of $h^i(\theta_l)$ in (5.11), it has a neglectable impact on the gradient and Hessian at the l-th position. This is because it results in a factor f_l^i which is close to 0. This is an analogy to the EM algorithm (Algorithm 8), where a low responsibility of cluster l for observation x_i results in x_i having no remarkable impact on estimating the Gaussian parameters of the l'th component (M-step), see Appendix B.

Equipped with the formulas for higher-order information derived in this section, we introduce a Riemannian Newton trust-region algorithm for fitting Gaussian mixture models in the following.

5.3 Riemannian Newton Trust-Region algorithm for GMMs

The explicit expression for the Riemannian Hessian stated in Theorem 5.7 allows us to apply Riemannian Newton-type algorithms for the fitting of Gaussian mixture models. However, the Riemannian Newton method comes with a couple of limitations explained in Section 2.2.4, one of which is that Newton's method is not globally convergent in case the Hessian is not positive definite in every point [105]. Since we cannot immediately see where the Riemannian Hessian (5.31) - (5.32) is positive definite, we need some safeguarding strategy like a trust-region approach. We consider a Riemannian Newton trust-region approach (Algorithm 3), where we use the Riemannian gradient and the Riemannian Hessian derived in the precedent section. For the subproblem (2.21), we use the truncated conjugate gradient method (Algorithm 4) with the inner product given by (5.25). In the following, we first consider theoretical convergence of R-NTR for our problem of fitting Gaussian mixture models and then discuss some practical considerations that turned out to be beneficial in the experiments conducted.

5.3.1 Convergence of R-NTR for Gaussian Mixture Models

To show the global convergence of the Riemannian Newton trust-region problem theoretically, we must ensure that certain regularity requirements are fulfilled, see Theorem 2.31. Let

$$\{\theta^t\}_{t\in\mathbb{N}_0} = \left\{\left((S_1^t,\dots,S_K^t),\eta^t\right)\right\}_{t\in\mathbb{N}_0}$$

be a sequence generated by Algorithm 3. By Theorem 5.5, the matrices S_j^t are bounded away from the boundary of $\mathbb{P}^{d+1}$ and remain in the interior of $\mathbb{P}^{d+1}$. In order to apply Theorem 2.31, we must further ensure that the matrices S_j^t do not get arbitrarily large, i.e. that for each $t = 0, 1, \dots$, there exists $0 < \tau^t < C$ such that

$$\|S_j^t\| \leq \tau^t \|\Psi\|, \tag{5.39}$$

where $\|\cdot\|$ is an arbitrary matrix norm on $\mathbb{R}^{(d+1)\times(d+1)}$, Ψ is as in (5.15) and $C > 0$ is a constant. Further, assume that there exists $\epsilon > 0$ such that

$$\alpha_j^t > \epsilon \tag{5.40}$$

for all $t = 0, 1, \dots$. With this assumption, convergence to a stationary point is ensured:

Theorem 5.8. *Consider the penalized reformulated objective $\hat{\mathcal{L}}_{\text{pen}}$ from (5.17). Let $\{\theta^t\}_{t\in\mathbb{N}_0}$ be a sequence generated by the Riemannian Newton trust-region Algorithm (Algorithm 3), where the quadratic subproblem is solved by truncated CG (Algorithm 4). Assume that there exists $C > 0$, $\epsilon > 0$ such that for each $t = 0, 1, \dots$ there exists $\tau^t < C$ such that*

$$\|S_j^t\| \leq \tau^t \|\Psi\|, \qquad \alpha_j^t > \epsilon \tag{5.41}$$

for all $j = 1, \dots, K$ *with* Ψ *as in* (5.15)*. Then, it holds that*

$$\lim_{t \to \infty} \operatorname{grad} \hat{\mathcal{L}}_{\text{pen}}(\theta^t) = 0. \tag{5.42}$$

Proof. The proof can be found in Appendix B.3. □

Note that the assumption (5.41) is a strong assumption and we cannot ensure convergence to a critical point by Theorem 5.8 if (5.39) and (5.40) are not fulfilled. However, for Gaussian mixture models, the S_j^t are covariance matrices related to the j-th Gaussian and thus depend on the concrete sample $x_1, \dots, x_n$. From a practical viewpoint, it is thus very unlikely that the matrices S_j^t become arbitrarily large such that (5.39) is deemed a reasonable assumption. The matrix Ψ is a prior belief about the covariance matrices of each cluster and a typical choice is that Ψ is set to a fraction of the overall sample covariance, see e.g. [66]. The assumption (5.40) can be considered as a lower bound of the proportion of each Gaussian, that is we assume that in each iteration, we have enough mass in each Gaussian. In the numerical experiments conducted, we observed numerical convergence to critical points, indicating that both (5.39) and (5.40) were fulfilled. Yet, to apply the theoretical global convergence result from Theorem 2.31, the assumption (5.39) and (5.40) are required and should be checked in each iteration.

The motivation for using a Riemannian Newton trust-region algorithm is its fast local convergence close to a local optimum. Theorem 2.32 states that the local convergence is superlinear close to an optimum if the Hessian satisfies a local Lipschitz continuity condition. For the problem of fitting Gaussian mixture models, we observe that the derived Hessian in Theorem 5.7 is continuously differentiable. Further, a local maximizer of the problem (5.21) is in the interior of $\overline{\mathcal{M}}_{GMM}$ according to Theorem 5.5 such that we can find a compact ball around a local nondegenerate maximizer θ^*. Thus, the extreme value theorem [77, Section 2.3] yields the local Lipschitz continuity such that all requirements of Theorem 2.32 are fulfilled. Therefore, if θ^* is a local nondegenerate maximizer of (5.21), we can find a neighborhood around θ^* where we have local superlinear convergence.

In the following, we discuss some practical considerations which turned out to be relevant for the implementation.

5.3.2 Practical Considerations

As outlined before in Chapter 4, tool support for Riemannian Optimization methods on the manifold of positive definite matrices is available. For the programming language

Python, the toolkit `pymanopt` [136] comes with an implementation of the Riemannian trust-region method and a truncated conjugate gradient method. Although the toolbox is especially helpful in case the gradient and Hessian is not available due to its support for automatic differentiation, the user can explicitly specify a formula for the gradient and Hessian, making it applicable for our framework with the Hessian (5.31)-(5.32). However, many terms in our problem occur both in the objective and the gradient as well as the Hessian. For this reason, we decided to implement a Riemannian Newton trust-region algorithm independent of the toolbox `pymanopt`, but followed the implementation framework of the `pymanopt` implementation.[1]

Choice of parameters for the R-NTR algorithm

The Riemannian trust-region algorithm (Algorithm 3) comes with a couple of hyperparameters that need to be set in advance. The parameter study in [53] and [34, Chapter 17] propose to set the thresholds for assessing the model quality close to their boundaries, that is ω_1 close to 0 and ω_2 close to 1. Further, if $\rho_t > \omega_2$, it is suggested by [53] to set the expansion parameter τ_2 higher than 2. Following these suggestions, we chose the hyperparameters of Algorithm 3 to be $\rho' = 0.1$, $\omega_1 = 1e-3$, $\omega_2 = 0.99$, $\tau_1 = 0.25$ and $\tau_2 = 3.5$. The initial trust-region radius Δ^0 is set by using the method suggested by [119] that is based on the model trust along the steepest-descent direction.

Stopping criterion of the R-NTR algorithm

A typical stopping criterion for the EM algorithm is if the increase in average log-likelihood between two subsequent iterates falls below some threshold [94, Section 4.8]. This cannot be immediately transferred to the Riemannian trust-region algorithm because if we reject a tentative next iterate, we remain in the same point $\theta \in \mathcal{M}_{GMM}$ thus yielding the same average log-likelihood (see Algorithm 3). For this reason, we only terminated the proposed Riemannian Newton trust-region algorithm if the difference between the average log-likelihood difference fell below the predefined threshold *and* we did not reject the tentative direction in this iteration. In addition, we observed that for badly scaled problems, we got stuck in regions where the trust-region radius was very small which resulted in tiny steps. For accepted tentative directions returned by the subproblem, this resulted in a termination of the algorithm although the returned

[1]At the time of the conduction of numerical experiments, the `pymanopt` version up to date (version 0.2.5) does not support such a reuse of terms for the calculation of objective, gradient and Hessian.

solution was far from a local optimum. Thus, we added the stopping condition that the Riemannian gradient in its Riemannian norm is close enough to zero. This condition is also a typical termination criterion of the classical (Riemannian) trust-region algorithm. We stop the R-NTR algorithm when all these conditions are fulfilled in order to ensure comparability with EM or when a prespecified number of iterations is reached.

Choice of preconditioner

In order to improve the convergence speed of the truncated conjugate gradient method, we used a preconditioner based on a LBFGS update as proposed by [100]: At an iteration t of Algorithm 3, we use the gradients computed in tCG (that is the residuals r_{n+1}) and store an inverse Hessian approximation via the LBFGS formula. The returned inverse Hessian approximation is then used for the minimization of the next subproblem $\hat{m}_{\theta^{t+1}}$. The use of such preconditioners has been suggested by [100] for solving a sequence of slowly varying systems of linear equations and gave a speed-up in convergence for our method. To be in line with a Riemannian framework, one would need to map the inverse Hessian approximations returned in every subproblem to the respective tangent space with the vector transport (4.17) to use it in the next iteration as a preconditioner. However, this would come with an additional high cost. Besides, the product manifold $\mathcal{M}_{GMM}$ is an open submanifold, for which reason the Hessian inverse approximation returned after solving the subproblem can be assumed to be a good approximation for a subsequent iterate as long as the step is small enough. For these reasons, we did not perform parallel transport on the inverse Hessian approximation before using it as a preconditioner in the next iteration. This performed well for the R-NTR method. For building the inverse Hessian approximations, we set the number of residual (gradient) storage equal to 5 for our experiments [105, Section 9.1].

5.4 Numerical Experiments

In this section, we provide numerical results of the proposed Riemannian Newton trust-region algorithm for Gaussian mixture models on both simulated and real-world data sets. We compare our method with the penalized Expectation Maximization algorithm (Appendix B.2, Algorithm 9) and with the Riemannian LBFGS method proposed by Hosseini and Sra in [65]. For the latter, we use the `MixEst` package [64] written by Hosseini and Mash'al in Matlab. We initialized all methods by running *k-means++* algorithm [11] and setting the initial values $\alpha_j^0, \mu_j^0, \Sigma_j^0$ to the sample proportion, the sample mean and the sample covariance, respectively. We stopped all methods when the difference in average penalized log-likelihood for two subsequent iterates fell below $1e-10$ (with the additional aforementioned conditions for R-NTR, see Section 5.3.2) or when the number of iterations exceeded 1500 for clustering and 3000 for density approximation. For the parameters of the penalization terms (5.14), we chose the parameters $\rho = \kappa = 0.01$, $\alpha = \beta = 1$, $\zeta = 1$, the matrix Λ was set to 0.01 times the sample covariance matrix of the $x_1, \ldots, x_n$ and λ was set to the sample mean according to [66].

All experiments in this chapter were performed in Python version 3.7 on an Intel Xeon CPU X5650 at 2.67 GHz with 24 cores and 20GB RAM.

Typical data mining tasks performed with Gaussian mixture models in practice are *clustering* and *probability density approximation*. We present results for both tasks in the following sections, Section 5.4.1 and Section 5.4.2.

5.4.1 Clustering with Gaussian Mixture Models

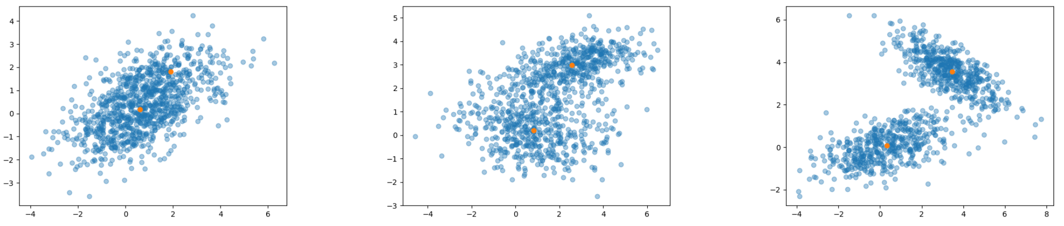
Using Gaussian mixture models for clustering is very popular as we can equip each single observation i with a probability r_{ij} that it belongs to a specific cluster j. When many data points have a non-negligible probability among more than one cluster, this means that we have a high overlap between the clusters and a high share of hidden information. In such a case, the Expectation Maximization algorithm converges slowly as outlined in Chapter 3. We tested our method for clustering problems with different levels of overlap with the simulation study described in what follows.

Simulation study

We simulate data following a Gaussian mixture model with different level of overlaps. For this, we consider the simulation design as proposed in [37, 65]. The K Gaussian distributions are chosen such that their means satisfy

$$\|\mu_i - \mu_j\|^2 \geq c \max_{i,j} (\mathrm{tr}(\Sigma_i), \mathrm{tr}(\Sigma_j)) \quad i, j = 1, \ldots, K, i \neq j,$$

where c models the degree of separation (see Figure 5.1) Additionally, a low *eccentricity* (or condition number) of the covariance matrices has an impact on the performance of Expectation Maximization [37], for which reason we also consider different values of eccentricity $e = \sqrt{\left(\frac{\lambda_{max}(\Sigma_j)}{\lambda_{min}(\Sigma_j)}\right)}$, as this is a measure of how much the data scatters. We followed the implementation of the `MixEst` toolbox [64] to generate the data.



(a) $c = 0.2$ (b) $c = 1$ (c) $c = 5$

Figure 5.1: Impact of the parameter c as a measure for cluster separation: low (5.1a), mid (5.1b) and high (5.1c) separation

We test our method on 20 and 40-dimensional data and an equal distribution among the clusters, i.e. we set $\alpha_j = \frac{1}{K}$ for all $j = 1, \ldots, K$. Although it is known that unbalanced mixing coefficients α_j result in slower EM convergence, this effect is less strong than the level of overlap [102]. For this reason, we here focus on balanced clusters.

We first take a look at the 20-dimensional data sets, for which we simulated $n = 1000$ data points for each parameter setting. In Table 5.1a, we show the results for very scattered data, that is $e = 1$. We see that, like predicted by literature, the Expectation Maximization converges slowly in such a case. This effect is even stronger with a lower separation constant c. The effect of the eccentricity becomes even more clear when comparing the results of Table 5.1a with Table 5.1b. Also the Riemannian algorithms converge more slowly for lower values of eccentricity e and separation levels c. However, they seem to suffer less from hidden information than Expectation Maximization. The proposed Riemannian Newton trust-region algorithm (R-NTR) beats the other methods in terms of runtime and number of iterations (see Figure 5.2a). The Riemannian LBFGS (R-LBFGS) method by [65] also shows faster convergence than EM, but the gain of second-order information available by the Riemannian Hessian is obvious. However, the R-LBFGS results created by the `MixEst` toolbox [64] show long runtimes compared to the other methods. We see from Figure 5.2b that the average penalized log-likelihood is slightly higher for R-LBFGS in some experiments. Still, the objective evaluated at the point satisfying the termination criterion is at a competitive level in all methods (see also Table 5.1a).

Table 5.1: Simulation results of 20 runs for dimensions $d = 20$, number of components $K = 5$ for different values of eccentricity

(a) eccentricity $e = 1$

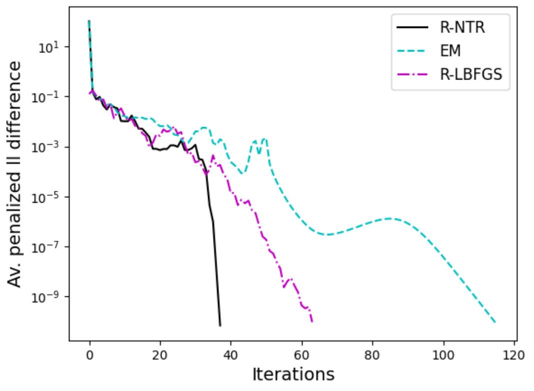
		EM	R-NTR	R-LBFGS
c=0.2	Iterations	295	79.4	113.4
	Mean time (s)	3.8	2.7	16.0
	Mean ALL	-42.64	-42.63	-42.63
	MSE weights	0.0014	0.0014	0.008
	MSE means	0.14	0.14	0.13
	MSE cov	2.27	2.5	2.28
c=1	Iterations	262	47.5	102.7
	Mean time (s)	3.7	2.1	14.3
	Mean ALL	-41.2	-41.21	-41.21
	MSE weights	0.009	0.010	0.008
	MSE means	0.23	0.22	0.24
	MSE cov	0.67	0.56	0.7
c=5	Iterations	208.8	54.2	92.4
	Mean time (s)	2.7	2.1	13.0
	Mean ALL	-36.98	-36.98	-36.99
	MSE weights	0.003	0.003	0.008
	MSE means	0.15	0.17	0.16
	MSE cov	9.81	7.1	10.19

(b) eccentricity $e = 10$

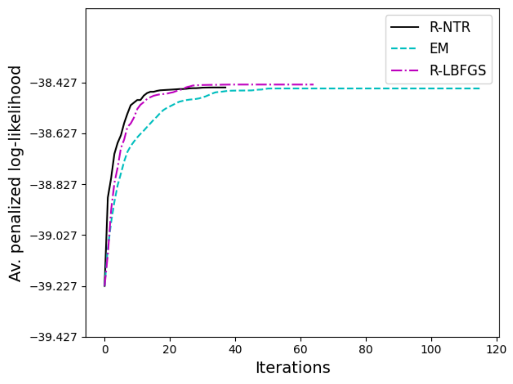
		EM	R-NTR	R-LBFGS
c=0.2	Iterations	66.2	16	33.2
	Mean time (s)	0.9	0.8	4.0
	Mean ALL	-60.06	-60.06	-60.07
	MSE weights	3e-05	3e-05	0.008
	MSE means	0.07	0.07	0.07
	MSE cov	0.31	0.23	0.31
c=1	Iterations	56.6	17.4	30
	Mean time (s)	0.7	0.8	3.6
	Mean ALL	-62.82	-62.82	-62.83
	MSE weights	3e-05	3e-05	0.008
	MSE means	0.09	0.09	0.09
	MSE cov	0.17	0.16	0.17
c=5	Iterations	43.1	14.7	29
	Mean time (s)	0.6	0.7	3.4
	Mean ALL	-61.04	-61.04	-61.05
	MSE weights	4e-05	4e-05	0.008
	MSE means	0.08	0.08	0.08
	MSE cov	0.13	0.14	0.13

When increasing the eccentricity (Table 5.1b), we see that the Riemannian methods still converge faster than EM in terms of number of iterations, but our method is not faster than EM in terms of runtime. This is because EM benefits from very low per-iteration costs and the gain in number of iterations is less strong in this case. However, we see that the Riemannian Newton trust-region method is not substantially slower. Furthermore, the average log-likelihood values (ALL) are more or less equal in all methods, so we might assume that all methods stopped close to a similar optimum. This is also underlined by comparable mean squared errors (MSE) to the true parameters from which the input data has been sampled from. In average, Riemannian Newton trust-region gives the best results in terms of runtime and number of iterations.

In Table 5.2a, we show results for dimension $d = 40$ and low eccentricity ($e = 1$) and the same simulation protocol as above (in particular, $n = 1000$). We observed that with our method, we only performed very few Newton-like steps and instead exceeded the trust-region within the tCG many times, leading to poorer steps (see also Figure 5.3b). One possible reason is that the number of parameters increases with d quadratically, that is in $\mathcal{O}(Kd^2)$, while at the same time we did not increase the number of observations $n = 1000$. If we are too far from a local optimum and the clusters are not well initial-



(a) Average penalized log-likelihood reduction

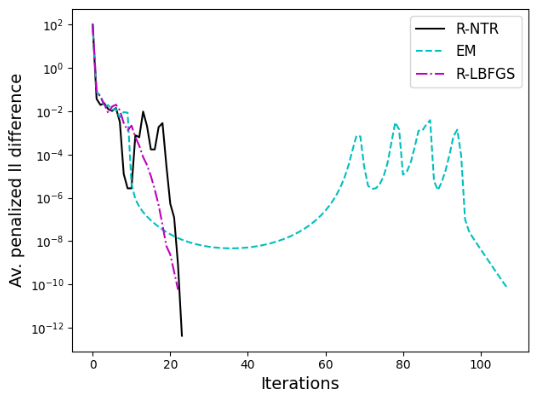


(b) Average penalized log-likelihood

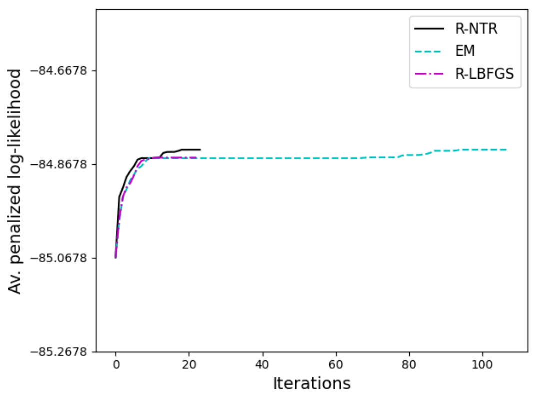
Figure 5.2: Average penalized log-likelihood reduction (a) and average penalized log-likelihood (b) for highly overlapping clusters: $d = 20$, $K = 5$, $e = 1$, $c = 0.2$.

ized due to few observations, the factor f_l^i in the Hessian (Theorem 5.7) becomes small, leading to large potential conjugate gradients steps (see Algorithm 4). Although this affects the E-step in the Expectation Maximization algorithm as well, the effect seems to be much severe in our method. To underline this, we show simulation results for a higher number of observations, that is, $n = 10.000$, in Table 5.2b with the same true parameters $\alpha_j, \mu_j, \Sigma_j$ as in Table 5.2a. As expected, the superiority in runtime of our method becomes visible: The R-NTR method beats EM with a factor of 4. Just like for the case of a lower dimension $d = 20$, the mean average log-likelihood and the errors are comparable between our method and EM, whereas R-LBFGS shows slightly worse results although it attains comparable runtimes to our method in this setting. We thus see that the ratio between number of observations and number of parameters must be large enough in order to benefit from the Hessian information in our method.

The simulation study performed shows that the use of the explicit formula for the Riemannian Hessian shows better runtimes and faster convergence than state-of the art methods for Gaussian mixture models if the number of observations n is large enough compared to the number of dimensions d. This effect is especially strong for settings where we have a high overlap between clusters, as in this case EM converges very slow. To further strengthen these findings, we also tested the R-NTR method on some real-world data sets.



(a) Average penalized log-likelihood reduction



(b) Average penalized log-likelihood

Figure 5.3: Average penalized log-likelihood reduction (a) and average penalized log-likelihood (b) for overlapping clusters: $d = 40$, $K = 5$, $e = 1$, $c = 1$.

Table 5.2: Simulation results of 20 runs for dimensions $d = 40$, number of components $K = 5$ and eccentricity $e = 1$ for different number of observations

(a) $n = 1000$ observations

		EM	R-NTR	R-LBFGS
c=0.2	Iterations	57.4	27.9	40.7
	Mean time (s)	1.3	2.3	6.7
	Mean ALL	-84.7935	-84.79	-84.7895
	MSE weights	0.00023	0.00023	0.008
	MSE means	0.104	0.09	0.104
	MSE cov	0.282	0.196	0.281
c=1	Iterations	61	29	48.6
	Mean time (s)	1.4	2.5	8.3
	Mean ALL	-82.3395	-82.3384	-82.3358
	MSE weights	0.0002	0.0002	0.008
	MSE means	0.076	0.084	0.076
	MSE cov	0.139	0.128	0.14
c=5	Iterations	81.8	28.8	49.2
	Mean time (s)	1.8	2.2	8.6
	Mean ALL	-92.4886	-92.4874	-92.4925
	MSE weights	0.00013	0.00013	0.008
	MSE means	0.08	0.095	0.08
	MSE cov	0.116	0.133	0.117

(b) $n = 10.000$ observations

		EM	R-NTR	R-LBFGS
c=0.2	Iterations	350.4	33.2	69.4
	Mean time (s)	53.621	12.455	15.449
	Mean ALL	-86.5717	-86.5718	-86.5731
	MSE weights	0.00043	0.00043	0.008
	MSE means	0.093	0.086	0.093
	MSE cov	0.207	0.195	0.206
c=1	Iterations	495.6	63.6	107.3
	Mean time (s)	79.955	20.739	23.153
	Mean ALL	-84.3783	-84.3779	-84.3797
	MSE weights	0.00062	0.00065	0.008
	MSE means	0.075	0.064	0.076
	MSE cov	0.075	0.038	0.075
c=5	Iterations	260.4	28.8	54
	Mean time (s)	42.6	10.434	11.692
	Mean ALL	-94.5592	-94.5591	-94.5603
	MSE weights	0.00012	0.00013	0.008
	MSE means	0.071	0.086	0.071
	MSE cov	0.045	0.053	0.045

Real-world data

We tested our method on some real-world data sets from *UCI Machine Learning repository* [46] besides the simulated data sets. For this, we normalized the data sets and tested the methods for different values of K.

Combined Cycle Power Plant Data Set [75]. In Table 5.3, we show the results for the combined cycle power plant data set. Although the dimension is moderate, we see that we can beat EM both in terms of runtime and number of iterations for $K = 5$, $K = 10$, $K = 15$ by applying the Riemannian Newton trust-region method. This underlines the results previously shown for simulation data. The gain by our method becomes even stronger when we consider a large number of components K where the overlap between clusters is large. We can reach a local optimum with our method in up to 15 times less iterations and a time saving of factor close to 4.

Table 5.3: Results of (normalized) combined cycle power plant data set for different number of components. Number of observations $n = 9568$, dimensions $d = 5$.

		EM	R-NTR	R-LBFGS
K = 2	Time (s)	0.40	0.63	2.38
	Iterations	56	19	34
	ALL	-4.24	-4.24	-4.24
K = 5	Time (s)	3.29	2.26	7.50
	Iterations	239	48	70
	ALL	-4.01	-4.01	-4.01
K = 10	Time (s)	31.72	4.28	23.40
	Iterations	1097	58	110
	ALL	-3.83	-3.82	-3.83
K = 15	Time (s)	28.27	6.79	35.77
	Iterations	677	67	111
	ALL	-3.75	-3.75	-3.75

MAGIC Gamma Telescope Data Set [20]. We also study the behaviour on a data set with higher dimensions and a larger number of observations with the MAGIC Gamma Telescope Data Set, see Table 5.4. Here, we can also observe a lower number of iterations in the Riemannian Optimization methods. Similarly to the combined cycle power plant data set, this effect becomes even stronger for a high number of clusters where the ratio of hidden information is large. Our method shows by far the best runtimes. For this data set, the average log-likelihood values are very close to each other except for $K = 15$ where the ALL is worse for the Riemannian methods. It seems that in this case, the R-NTR and the R-LBFGS methods end in different local maxima than the EM. However, for all of the methods, convergence to global maxima is theoretically not ensured and for all methods, a globalization strategy like a split-and-merge approach [84] might improve the final ALL values. As the Magic Gamma telescope data set is a classification data set with 2 classes, we further report the classification performance in Table 5.5a. We see that the geodesic distance defined on the manifold and the weighted mean squared errors (wMSE) are comparable between all three methods. In Table 5.5b, we also report the adjusted rand index being a cluster validation measure [70] for all methods. Here, a value of 1 between two methods means that all data points are assigned to the same clusters. Although the clustering performance is very low compared to the true class labels (first row), we see that it is equal among the three methods.

Table 5.4: Results of (normalized) magic gamma telescope data set for different number of components. Number of observations $n = 19020$, dimensions $d = 11$.

		EM	R-NTR	R-LBFGS
K = 2	Time (s)	1.18	0.57	1.52
	Iterations	30	6	17
	ALL	-7.81	-7.81	-7.81
K = 5	Time (s)	3.96	1.32	5.37
	Iterations	65	9	34
	ALL	-6.53	-6.53	-6.53
K = 10	Time (s)	36.00	6.99	20.78
	Iterations	293	34	77
	ALL	-6.02	-6.02	-6.02
K = 15	Time (s)	56.24	12.61	50.52
	Iterations	354	38	115
	ALL	-5.39	-5.54	-5.51

We show results on additional real-world data sets in Appendix B.

Table 5.5: Model quality for (normalized) magic gamma telescope data set for $K = 2$

(a) geodesic distance and weighted MSE

	EM	R-NTR	R-LBFGS
distance	4.783910	4.783934	4.783922
wMSE weight	0.000034	0.000034	0.000034
wMSE mean	1.883762	1.883765	1.883759
wMSE cov	8.214824	8.214840	8.214828

(b) adjusted rand index

	truth	EM	R-NTR	R-LBFGS
truth	1	0.06	0.06	0.06
EM		1	1.00	1.00
R-NTR			1	1.00
R-LBFGS				1

5.4.2 Density Estimation with Gaussian Mixture Models

Besides the task of clustering (multidimensional) data, Gaussian mixture models are also well-known to serve as probability density estimators for unknown smooth density functions [123]. We describe the basic idea of probability density estimation in the following. We are given observations $x_1, \dots, x_n$ and the goal is to find a probability density function that fits the data well. For the problem of probability density estimation, the true distribution family from which the observations $x_1, \dots, x_n$ are drawn is unknown and a generative model is used. Typical approaches of probability density estimation are (nonparametric) kernel density estimation [101, Section 14.7] or using a parametric approach like a Gaussian mixture model with sufficient components [123].

In this section, we consider Gaussian mixture models for the purpose of probability density estimation. For fitting the parameters of a Gaussian mixture model for given observations $x_1, \dots, x_n$, we consider the Riemannian Newton trust-region method, the EM algorithm and the R-LBFGS method and compare the approximation power of the resulting Gaussian mixture models with each other.

To analyze this, observations $x_1, \dots, x_n$ were generated according to a bivariate Beta-Gamma distribution (i.e. $d = 2$) with parameters $\alpha_{Beta} = 0.5$, $\beta_{Beta} = 0.5$, $a_{Gamma} = 1$, $\beta_{Gamma} = 1$. The joint distribution was characterized by a Gaussian copula [74, Section 1.3]. The resulting density function surface is visualized in Figure 5.4.

We simulated 100 data sets each containing $n = 1000$ realizations of the Beta(0.5,0.5)-Gamma(1, 1) distribution. For each of the data sets, we fitted a Gaussian mixture model with the Riemannian Newton trust-region method, the EM and the R-LBFGS method. To investigate the individual approximation power, we considered different values of K and compared the (approximated) root mean integrated squared error (RMISE) to compare the model quality obtained with the individual optimizers against each other.

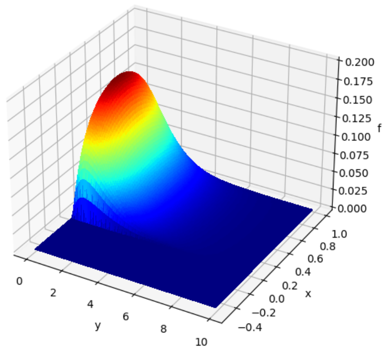


Figure 5.4: Probability density of bivariate Beta(0.5,0.5)-Gamma(1, 1) distribution.

The RMISE is a measure of approximation power for a probability density estimator $\hat{f}$ to a known probability density function f and is given by [54]

$$RMISE(\hat{f}) = \sqrt{\mathbb{E}\left(\int \left(f(x) - \hat{f}(x)\right)^2 dx\right)}, \tag{5.43}$$

where f denotes the underlying true density, i.e. in our application Beta(0.5,0.5)-Gamma(1, 1), and $\hat{f}$ the density approximator (GMM). In order to compute the RMISE (5.43), we consider N equidistant grid points g_r and compute [54]

$$RMISE(\hat{f}) \approx \sqrt{\frac{1}{N}\sum_{r=1}^{N}\left(f(g_r) - \hat{f}(g_r)\right)^2 \delta_g^2}$$

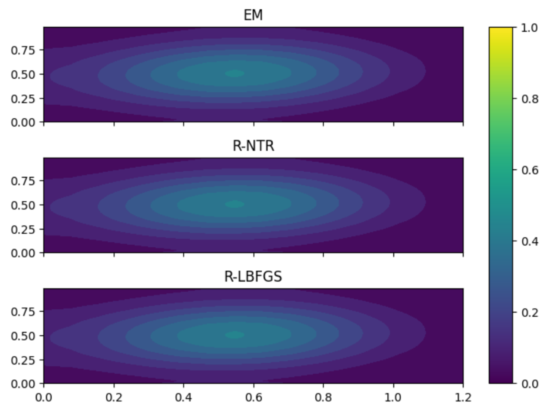
with grid width δ_g. For our simulation study, we chose 16384 grid points in the box $[0, 5] \times [0, 10]$. We show the results in Table 5.6, where we fit the parameters of the GMM by the method introduced in this thesis (R-NTR) and compare against GMM approximations where we fit the parameters with EM and R-LBFGS, respectively.

We observe that the RMISE shows comparable values for all methods. Just as for the clustering results in Subsection 5.4.1, we have much lower runtimes for R-NTR and a much lower number of total iterations. This is a remarkable improvement especially for a larger number of components. We also observe that in all methods, the mean average log-likelihood (ALL) of the training data sets with 1000 observations attains higher values with an increasing number of components K. This supports the fact that the approximation power of GMMs for a continuous density function is expected to become higher if we add additional Gaussian components [123, 52]. On the other hand,

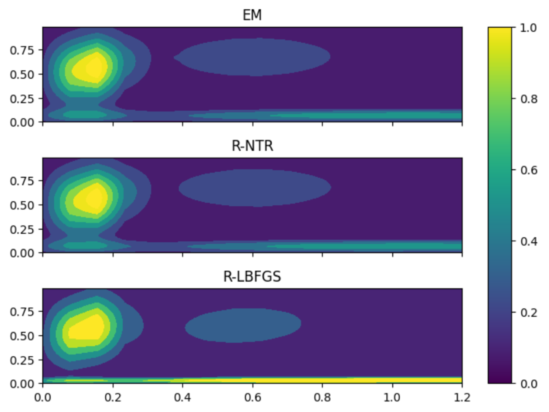
Table 5.6: Simulation results averaged over 100 simulation runs for approximation of a Beta(0.5,0.5)-Gamma(1, 1) distribution by a Gaussian mixture models with different values of K. Parameter Estimation by EM, R-NTR and R-LBFGS.

		EM	R-NTR	R-LBFGS
K=2	Mean RMISE	0.00453	0.00453	0.00453
	SE RMISE	0.00017	0.00017	0.00017
	Iterations	113	27.9	25.9
	Mean time (s)	0.2002	0.1532	0.9877
	Mean ALL	-1.53103	-1.52953	-1.53103
K=5	Mean RMISE	0.00565	0.00565	0.00567
	SE RMISE	0.00028	0.00024	0.00025
	Iterations	346.5	47	72.9
	Mean time (s)	1.0893	0.5202	6.7941
	Mean ALL	-1.22852	-1.22108	-1.22912
K=10	Mean RMISE	0.00639	0.00643	0.00643
	SE RMISE	0.00025	0.00027	0.00029
	Iterations	623.1	56.8	112.2
	Mean time (s)	3.4226	1.7108	20.4754
	Mean ALL	-1.06821	-1.02844	-1.06781
K=15	Mean RMISE	0.00667	0.00669	0.0067
	SE RMISE	0.00026	0.00027	0.00026
	Iterations	791.5	62.9	135.2
	Mean time (s)	6.1649	2.5701	37.6
	Mean ALL	-1.02677	-0.95907	-1.02604
K=20	Mean RMISE	0.00681	0.00683	0.00685
	SE RMISE	0.00026	0.00025	0.00026
	Iterations	874.8	66.9	144.6
	Mean time (s)	8.7694	3.3773	55.6337
	Mean ALL	-1.02029	-0.9202	-1.02056

the RMISE (which is not based on the training data) increased in our experiments with larger K's. This means that we are in a situation of overfitting [101, Section 1.4.7]. The drawback of overfitting is well-known for EM [8] and we also observed this for the R-NTR and the R-LBFGS methods. However, the RMISE are comparable and so none of the methods outperforms another substantially in terms of overfitting. This can also be seen from Figure 5.5 showing the distribution of the pointwise errors for $K = 2$ and $K = 5$. Although the R-LBFGS method shows higher error values on the boundary of the support of the distribution for $K = 5$, the errors show similar distributions among the three methods at a comparable level. We propose methodologies such as cross validation [101, Section 1.4] or applying a split-and-merge approach on the optimized parameters [84] to address the problem of overfitting.



(a) $K = 2$



(b) $K = 5$

Figure 5.5: Contours of pointwise root mean squared error (RMSE) for density approximation via GMMs of the Beta(0.5,0.5)-Gamma(1, 1) distribution.

5.4.3 Summary of Numerical Results

We tested our proposed method, the Riemannian Newton trust-region algorithm with the Riemannian Hessian derived in Theorem 5.7 for both a clustering and a density approximation task. For the experiments conducted, we see that our method is an effective alternative for the widely used Expectation Maximization algorithm and the Riemannian LBFGS method proposed in [65] for 20- and 40- dimensional data. Especially in settings where there is low separation power between the K Gaussians (high value of K and/or low eccentricity), we gain a time saving of up to factor 10 (Table 5.2b) which is remarkable in spite of the very low per-iteration costs of the EM algorithm. For settings where there is only very few overlap between the K Gaussians, the EM can easily assign each data point to a cluster in the E-step and the EM converges fast as expected following the discussion in Chapter 3. We also observed that the benefit of the Riemannian approach cannot beat EM in such settings, see Table B.2a. Still, for such a setting, the overall runtimes of all investigated methods are on a low level and hence the time loss by the R-NTR method is less severe. With regard to the discussions about the disadvantages of proposed alternatives to EM in Section 5.1, Riemannian approaches can keep up with the EM algorithm and can thus be considered a reasonable alternative.

5.5 Concluding Remarks and Future Research Directions

Gaussian mixture models are widely used for both clustering problems as well as probability density approximation tasks in many application fields [94, 95]. A Gaussian mixture model is described by a probability distribution consisting of K Gaussian distributions and the problem of fitting the model is given by finding the mixing proportions and parameters of the Gaussian distributions. This chapter dealt with finding optimal parameters of a Gaussian mixture model with Riemannian optimization. For this, we used the Riemannian formulation of the corresponding optimization problem as introduced by Hosseini and Sra in [65, 66] which exploits the Riemannian geometry of covariance matrices appearing in the Gaussian distributions of the K Gaussians (Section 5.2). A contribution of this thesis is the derivation of the Riemannian Hessian in Section 5.2.2. Based on the Riemannian Hessian for Gaussian mixture models, the Riemannian Newton trust-region algorithm was proposed in Section 5.3 and important problem-specific convergence results have been shown (Section 5.3.1). In Section 5.3.2, we also gave practical guidance for the Riemannian Newton trust-region method that turned out to be beneficial for the numerical tests presented in this thesis. The novel approach for Gaussian mixture models by considering a Riemannian Newton trust-region algorithms was tested on various different data sets in Section 5.4. We observed that in the case of a high share of hidden information, the Riemannian Newton trust-region algorithm showed remarkably faster results than the investigated alternative approaches both in terms of number of iterations and runtime. The following future research directions based on the work presented within this thesis would be interesting to investigate:

- *Riemannian optimization for Gaussian mixture models with higher dimension.* In the numerical experiments studied in this thesis, we considered observations of dimensions $d = 20$ and $d = 40$ and investigated the Riemannian Newton trust-region algorithms with respect to these dimensions. It would be interesting to explore whether the Riemannian Newton trust-region algorithm shows superior results compared to state of the art optimizers in higher dimensions d. When increasing the dimension d, the risk of covariance singularity in Σ_j get higher. An approach to avoid this singularity and the curse of dimensionality is to impose a special structure on the covariance matrices Σ_j like using a principal component analysis or factor analyzers model [95]. Incorporating such structural assumptions into the Riemannian framework is an important future research direction. Another approach to give the covariance matrices Σ_j more structure is to assume uncorrelatedness of variables within a Gaussian component. This results in zero entries at the respective positions of Σ_j. As an extreme example, we could impose $\Sigma_j = \operatorname{diag}(\sigma_r^2)$, $r = 1, \ldots, d$ which reduces the complexity from $\mathcal{O}(d^2)$ to a complexity of $\mathcal{O}(d)$. A straightforward extension of the Riemannian framework could be to model the covariance matrices Σ_j as block diagonal matrices of smaller covariance matrix blocks.

- *Reducing computational costs of evaluating the Riemannian gradient and Hessian.* In this chapter, we derived the Riemannian Hessian for Gaussian mixture models in Theorem 5.7. The computational costs of evaluating the Riemannian Hessian highly depends on the number of observations n as we need to sum over all observations. As outlined in Section 5.2.2, the contribution of the i-th observation to the gradient and Hessian of the l-th component is neglectably low if observation i is unlikely of originating from the l-th component. Thus, for a larger number of observations n, it might be helpful to first initialize the component membership e.g. by running $kmeans++$ or performing some EM steps for Gaussian mixture models in order to get initial component membership probabilities. Then, only observations i that show a remarkable probability of arising from the l-th component could be used for evaluating the Riemannian gradient and the Riemannian Hessian. Such an approximation of the Riemannian gradient and Riemannian Hessian might yield a big saving of computational costs. It is believed that many data points show a high probability of arising from a few clusters $\tilde{K} < K$, even for highly overlapping settings.

- *Consideration of higher number of observations n.* In this thesis, we considered data sets of a moderate number of observations n. The outlined idea of considering only a subset of data points for an approximation of the Riemannian gradient and Riemannian Hessian could give a speed-up for the Riemannian trust-region method for higher values of n. In the work [66], Hosseini and Sra have introduced a Riemannian stochastic gradient descent method which they test for larger n. A thorough comparison of the Riemannian trust-region method with possibly approximate Riemannian gradient and Hessian with the Riemannian stochastic gradient descent method proposed in [66] could reveal potential advantages and limitations of both methods for large values of n.

In this chapter, we considered Gaussian mixture models which can be used for probabilistic model-based clustering. The introduced Riemannian approach for fitting the parameters proved efficient for the model. In the following chapter, we consider Riemannian optimization for another statistical model, that is the *linear mixed model.* In contrast to the Gaussian mixture model, the clustering structure is known beforehand and incorporated into the modeling of linear relationships between variables.

CHAPTER 6

Riemannian Optimization for Variance Estimation in Linear Mixed Models

In many classical statistical models, we assume that data are drawn from the same setting/ population, that is we assume observations are independent and identically distributed (iid data assumption). A typical classical statistical model used for prediction is the linear regression model, where we model a linear relationship between the observed *response* (dependent variable) $y_i \in \mathbb{R}$ and the *covariate vector* (independent variables) $x_i \in \mathbb{R}^p$ with homoscedastic and independent errors, that is

$$y_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_{p-1} x_{p-1i} + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2) \tag{6.1}$$

for observations (x_i, y_i), $i = 1, \ldots, n$, see [128, Section 4.1]. Here, the parameter $\beta = (\beta_0, \ldots, \beta_{p-1}) \in \mathbb{R}^p$ and the residual variance σ^2 are unknown and have to be estimated. The parameter β_0 is called the *intercept* and the parameters β_i, $i = 1, \ldots, n$ are *fixed effects* or *regression coefficients* that are associated with the slopes of the variables $x_{1i}, \ldots, x_{p-1i}$. In vector notation, the linear regression model (6.1) reads

$$y = X\beta + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2 I_n), \tag{6.2}$$

where $y = (y_1, \ldots, y_n)$, $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)$. $X \in \mathbb{R}^{n \times p}$ is the matrix resulting from stacking the variables $x_i = (x_{1i}, \ldots, x_{p-1i})$ on top of each other and I_n is the identity matrix of size n [128, Section 4.1].

The linear regression model (6.2) is well studied and frequently used in practice in different application fields. Yet, it comes with strong assumptions like the homoscedasticity and independence of errors which is not fulfilled in many real-world settings. Often, data are collected from different *clusters* reflecting a specific sample of the overall population. Such a grouped structure occurs for example for repeated measures design in clinical research, where the same variables are measured repeatedly [40, Chapter 1]. In such settings, we may reasonably suppose that the iid assumption does not hold true and the linear regression model (6.2) is deemed inappropriate [111, Section 1.4.1]. A way to cope with a clustered dependence structure in linear models is to consider *linear mixed models* or *linear mixed effects (LME) models.* These are an appropriate model for grouped data capturing two sources of variation: a variation within and a variation between clusters (groups) [40, Section 1.1]. This chapter deals with the fitting of linear mixed models.

The chapter is structured as follows. In Section 6.1, we introduce the linear mixed model formally. Section 6.2 deals with parameter estimation in the linear mixed model which can be divided into the estimation of fixed and random effects parameters and the variance parameters involved in the model. Further, we review approaches to estimate the variance parameters. In Section 6.3, we formulate the estimation of the variance parameters of a linear mixed model as a Riemannian optimization problem. We derive the Riemannnian gradient and the Riemannian Hessian for the *REML log-likelihood* objective. In Section 6.4, we compare the Riemannian approach with the approach from the popular `lme4` package [44] numerically. We close the chapter with a summary and a discussion about future research directions in Section 6.5.

6.1 The Linear Mixed Model

As previously outlined, modeling a linear relationship in data with the classical linear regression model (6.2) might not be appropriate in case we have clustered data because the iid property might be violated. As a remedy, we consider linear mixed models. Linear mixed models extend the classical linear regression model in the following way: Based on the linear relationship between covariates x_i and a response y_i with the *fixed effects parameter* β, we add an additional term that represents the variability in the model introduced by considering different group categories in the data, i.e. the *between groups variability* [40, Section 1.1]. This variability is modeled by so-called *random effects.* The random effects can be considered as a realization from a random variable that models the deviation of a specific group category from the overall population [40, Section 1.2]. Here, it is assumed that the group categories available in the data are sampled at random from a common population and the random effects can be interpreted as surrogates of

incomplete group-specific measurements [111, Section 1.1].
In order to model data by a linear mixed model, there must be at least one categorical variable capturing the group structure [91]. We assume that we have $K \in \mathbb{N}$ such categorical variables of interest in our data set. We call these categorical variables *grouping factors* according to [43] and denote them by $\mathcal{B}_j$, $j = 1, \dots, K$. Further, we assume that M_j *levels* (groups, clusters) are sampled from $\mathcal{B}_j$, that is in an experiment, we observed M_j levels for a grouping factor $\mathcal{B}_j$. Depending on the design of the experimental study, we can have *nested* or *crossed* grouping factor designs describing the dependence structure of the grouping in the data. According to [43], we say that a grouping factor $\mathcal{B}_1$ is *nested* within a grouping factor $\mathcal{B}_2$ if each of the M_1 levels of $\mathcal{B}_1$ occurs in conjunction with one and only one level of $\mathcal{B}_2$. In particular, this means that $M_1 \geq M_2$ and $M_1 = M_2$ occurs only if $\mathcal{B}_1$ and $\mathcal{B}_2$ are identical. A collection of grouping factors $\mathcal{B} = \{\mathcal{B}_1, \dots, \mathcal{B}_K\}$ is said to be *strictly nested* if, when ordered to non-decreasing numbers of levels, each factor is nested within its successor. In contrast, grouping factors $\mathcal{B}_1$ and $\mathcal{B}_2$ are said to be *crossed* if every level of $\mathcal{B}_1$ occurs in conjunction with every level of $\mathcal{B}_2$ [43]. Apart from completely crossed designs, linear mixed models also allow for *partially crossed designs* which is a mixture of nested and crossed grouping factor interactions. We underline the interaction of different grouping factors by the following example. We consider industrial machines and are interested in predicting the defect based on different variables captured by sensors installed in the machines. For this, we sample data from machines from the same type but from M_1 different manufacturers, that is we have a grouping factor $\mathcal{B}_1$ representing the manufacturer with M_1 levels in the data. We add another grouping factor $\mathcal{B}_2$ representing the operator working with the machines. Assume we have M_2 operators who are working with machines from all manufacturers. Then, the grouping factors $\mathcal{B}_1$ (manufacturer) and $\mathcal{B}_2$ (operator) are crossed. To extend the model, we add a third grouping factor $\mathcal{B}_3$: each of the M_2 operators is trained by exactly one of M_3 instructors, where $M_3 \leq M_2$. Then, the grouping factor $\mathcal{B}_2$ (operator) is nested within grouping factor $\mathcal{B}_3$ (instructor). This is an example of a partially crossed design, that is we have both crossed and nested interactions.

The flexibility of modeling different dependence structures in grouping factors make linear mixed models a powerful tool to analyze data from complex study designs like repeated measures, blocked or multilevel data designs or longitudinal data [56]. Applications can be found in the field of neuroscience [55], psychology [43], statistical image analysis [40, Chapter 12] and many others.

We state the linear mixed model and explain the single components in the following. For this, we follow the notations of [56] and [40, Section 2.2]. In the following, we assume that both the random effects and the residual errors follow a Gaussian distribution and the residual errors are independent and identically distributed. Further, we here consider the setting where the response variable y is continuous. *Generalized linear mixed models*, where the response is not Gaussian, are beyond the scope of this thesis, we refer to [40] and [111] for an overview of such an extension.

The linear mixed model is given by the relationship

$$y = X\beta + Zb + \varepsilon. \tag{6.3}$$

The single components of (6.3) are explained in the following:

- $y \in \mathbb{R}^n$ is the *response* or dependent variable of the model, where n is the number of observations.
- The *fixed effects design matrix* $X \in \mathbb{R}^{n \times p}$ contains information about the *fixed effects variables* and possibly a fixed intercept. The design matrix $X \in \mathbb{R}^{n \times p}$ is specified by the user and is thus known.
- The *fixed effects parameter* $\beta \in \mathbb{R}^p$ describes the impact of the fixed effects variables onto the response. Similar to the classical linear regression model, β is not known beforehand and needs to be estimated.
- The *random effects design matrix* $Z \in \mathbb{R}^{n \times q}$ describes the *random effects variables.* If $K \in \mathbb{N}$ is the number of grouping factors, the matrix Z can be expressed by K known grouping factor specific design matrices $Z^{(j)}$, $j = 1, \ldots, K$, that is
$$Z = (Z^{(1)}, \ldots, Z^{(K)}),$$
where $Z^{(j)} \in \mathbb{R}^{n \times \tilde{q}_j}$, $q = \sum\limits_{j=1}^{K} \tilde{q}_j$ for $\tilde{q}_j \geq 1$.
- The *random effects parameter* $b = \left(b^{(1)}, \ldots, b^{(K)}\right) \in \mathbb{R}^q$, $b^{(j)} \in \mathbb{R}^{\tilde{q}_j}$ is a random variable with zero mean. Here, the *random effects parameter for group j* denoted by $b^{(j)}$ is associated with the grouping factor $\mathcal{B}_j$. It can be interpreted as the deviation introduced by considering a specific level/group compared to the overall population. This can be a deviation in location (random intercept) or a deviation

in slope (random slope) of a specific variable. We assume that b follows a Gaussian distribution, that is

$$b \sim \mathcal{N}\left(0, \tilde{G}\right), \tag{6.4}$$

where $\tilde{G} \in \mathbb{R}^{q \times q}$. The covariance matrix $\tilde{G}$ needs to be estimated in real-world applications. Usually, we do not need to estimate all $q(q+1)/2$ entries of the matrix $\tilde{G}$, but use a parameterization $\tilde{G} = \tilde{G}(\gamma)$ with $\gamma \in \mathbb{R}^r$, where $r \ll q(q+1)/2$ [56].

- The residual error $\varepsilon \in \mathbb{R}^n$ is a random variable. We assume a Gaussian distribution with zero mean and shared variance σ^2, that is

  $$\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n),$$

  where I_n is the identity matrix of size n. The parameter σ^2 needs to be estimated in applications.

With the linear mixed model (6.3), we can identify the marginal distribution of the response y as [56]

$$y \sim \mathcal{N}(X\beta, \sigma^2 H), \tag{6.5}$$

where

$$H = ZGZ^T + I \text{ and } G = \frac{1}{\sigma^2}\tilde{G}. \tag{6.6}$$

Some remarks with regard to the linear mixed model (6.3) are in order:

(i) The scaling of the random effects covariance matrix $G = 1/\sigma^2 \tilde{G}$ results in an easier expression of the profiled log-likelihood which will be explained in Section 6.2.3 and is thus often to be preferred [56], [40, Section 2.2].

(ii) With the linear mixed model (6.3), we can specify various random effects structures with the covariance matrix G. However, we assume the residual errors (within-group errors) to be iid, that is $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ but in some applications it is desirable to allow for heteroscedascity or correlation for the within-group errors. An according extension to the classical linear mixed model (6.3) can be found by replacing the assumption $\varepsilon \sim \mathcal{N}(0, \sigma^2 I)$ by the assumption $\varepsilon \sim \mathcal{N}(0, R(\omega))$ for a covariance matrix $R(\omega) \in \mathbb{R}^{n \times n}$ with $\omega \in \mathbb{R}^l$, $l \ll n(n+1)/2$ [56]. For an overview of different structures of $R(\omega)$, we refer to [111, Chapter 5].

(iii) The random effects design matrices $Z^{(j)}$ are typically constructed such that the cross-product ${Z^{(j)}}^T Z^{(j)} \in \mathbb{R}^{\tilde{q}_j \times \tilde{q}_j}$ is block-diagonal for each $j = 1, \dots, K$. For a detailed guide on how to construct $Z^{(j)}$ for a grouping variable $\mathcal{B}_j$, we refer to [15].

(iv) Modeling two nested grouping factors $\mathcal{B}_1$, $\mathcal{B}_2$, where $\mathcal{B}_1$ is nested in $\mathcal{B}_2$ via the design matrix Z can be achieved by combining the M_1 levels and M_2 levels with each other, i.e. we introduce a new grouping variable with $M_1 M_2$ levels reflecting the co-occurrence of levels from the first and the second grouping factor. Such an approach is used in the popular `lme4` package [44]. Thus, we do not distinguish between nested and crossed effect interactions in the following.

(v) The random effects covariance matrix $G \in \mathbb{R}^{q \times q}$ in (6.6) is a block-diagonal matrix with K matrices G_j on the diagonal. Here, G_j refers to the j-th grouping effect and is of size $\tilde{q}_j \times \tilde{q}_j$. We have

$$G = \begin{pmatrix} G_1 & & \\ & \ddots & \\ & & G_K \end{pmatrix}, \qquad \text{where } G_j = \begin{pmatrix} \Psi_j & & \\ & \ddots & \\ & & \Psi_j \end{pmatrix}, \tag{6.7}$$

that is G_j is itself a block-diagonal matrix with M_j blocks and the same smaller covariance matrix $\Psi_j \in \mathbb{R}^{q_j \times q_j}$ on the diagonal. Here, $\tilde{q}_j = M_j q_j$ for some $q_j \geq 1$ [15]. The structure of G_j reflects the assumption that each level in a grouping factor is drawn from the same distribution (levels are sampled from the same population) and that the $b^{(j)}$ are independent of each other.

6.2 Parameter Estimation in the Linear Mixed Model

In real-world experiments, we only observe realizations of the response y, of the fixed effects design matrix X and the random effects design matrix Z. Thus, we need to estimate the fixed effects parameter β, the random effects parameter b and the variance parameters σ^2 and G. We first examine the joint estimation of the fixed effects parameter β and the random effects b for given variance parameters σ^2, G in Section 6.2.1 and then consider the estimation of the *variance parameters* σ^2 and G in Section 6.2.2 according to [56].

6.2.1 Estimation of Fixed and Random Effects Parameters

Many methods have been proposed for estimating the fixed effects parameters and random effects parameters at the same time, see [56] for an overview. We here follow the approach by Henderson as proposed in [60]. Henderson models the joint distribution of the response y and the random effects parameter b as

$$(b, y) \sim \mathcal{N}(a, A), \tag{6.8}$$

where

$$a = \begin{pmatrix} 0 \\ X\beta \end{pmatrix} \in \mathbb{R}^{q+n}$$

and the covariance matrix A is given by

$$A = \sigma^2 \begin{pmatrix} G & GZ^T \\ ZG & H \end{pmatrix} \in \mathbb{R}^{(q+n)\times(q+n)}.$$

Thus, the marginal distribution of y has the form (6.5), i.e. $y \sim \mathcal{N}(X\beta, \sigma^2 H)$. The approach suggested by [60] considers the log-joint distribution of (y, b) given by (6.8) and maximizes it with respect to β and b. The log-joint distribution $\log(p_{\mathcal{N}}(\,\cdot\,; a, A))$ of (y, b) reads [56]

$$\log(p_{\mathcal{N}}((y,b); a, A)) = -\frac{1}{2}\Bigg((n+q)\log\sigma^2 + \log\det(G) + \frac{(y-X\beta)^T(y-X\beta)}{\sigma^2} + \frac{1}{\sigma^2}(b^T(ZZ^T + G^{-1})b - 2(y-X\beta)^T Zb)\Bigg).$$

For maximizing the log-likelihood $\log(p_{\mathcal{N}}((y,b); a, A))$ with respect to β and b, we need to solve the *mixed model equations* (MME)

$$\begin{pmatrix} X^TX & X^TZ \\ Z^TX & Z^TZ + G^{-1} \end{pmatrix} \begin{pmatrix} \beta \\ b \end{pmatrix} = \begin{pmatrix} X^Ty \\ Z^Ty \end{pmatrix} \tag{6.9}$$

for the vector $(\beta, b) \in \mathbb{R}^{p+q}$ resulting from the first-order conditions, see [60, 56]. Solving the MME for β and b results in the following expressions specified in Theorem 6.1.

Theorem 6.1. *[56, Lemma 1] The solutions of β and b from solving the MME* (6.9) *for known G and σ^2 are given by*

$$\hat{\beta} = (X^TH^{-1}X)^{-1}X^TH^{-1}y, \tag{6.10}$$

$$\tilde{b} = GZ^TH^{-1}(y - X\beta), \tag{6.11}$$

where $H = ZGZ^T + I$. The covariance matrices $\mathrm{Cov}(\hat{\beta})$, $\mathrm{Cov}(\tilde{b})$ *of $\hat{\beta}$, $\tilde{b}$, respectively, are given by*

$$\mathrm{Cov}(\hat{\beta}) = \sigma^2(X^TH^{-1}X)^{-1},$$

$$\operatorname{Cov}(\tilde{b}) = \sigma^2 G Z^T P(H) Z G,$$

where

$$P(H) = H^{-1} - H^{-1}X(X^T H^{-1} X)^{-1} X^T H^{-1}. \tag{6.12}$$

Proof. A proof can be found in [56]. □

The predictor $\tilde{b}$ is known as the *best linear unbiased predictor (BLUP)*, see [40, Section 3.7], [56]. The expressions in Theorem 6.1 assume that G (and thus H) and σ^2 are known, but in applications they are usually not. In this case, we replace σ^2, G (H) by their estimators $\hat{\sigma}^2$, $\hat{G}$ ($\hat{H}$) resulting from methods like maximum likelihood estimation [56].

In the following, we consider the estimation of the variance parameters σ^2, G.

6.2.2 ML and REML Estimation

An approach to estimate the parameters σ^2 and G is maximum likelihood (ML) estimation and residual maximum likelihood (REML) estimation.

ML Estimation

The marginal distribution of y is given by (6.5), hence the log-likelihood function l_{ML} of y reads [56]

$$l_{ML}\left((\beta, \sigma^2, G)\right) = -\frac{1}{2}\left(n\left(\log(2\pi\sigma^2)\right) + \log\det(H) + \frac{(y - X\beta)^T H^{-1}(y - X\beta)}{\sigma^2}\right), \tag{6.13}$$

where $H = ZGZ^T + I$, see Section 3.1.

The maximum likelihood approach with the objective (6.13) is known to underestimate the variance components, that is it is *biased downwards* [56], [111, Section 2.2.5]. This can be led back to the fact that the maximum likelihood estimation via (6.13) does not take into account the degrees of freedom lost in estimating the fixed effects [40, Section 2.2.5]. A surrogate for this downward biasedness is to consider the *residual maximum likelihood* (REML) as an objective function.

REML Estimation

The *residual maximum likelihood approach* (sometimes called *restricted maximum likelihood* [56]) maximizes the log-likelihood function for the residual vector $\tilde{\varepsilon} := y - X\hat{\beta}$, where $\hat{\beta}$ is given by the expression (6.10), see [56], [40, Section 2.2.5].

The REML log-likelihood function is given by [56]

$$\begin{aligned} & l_R(\sigma^2, G) \\ &= -\frac{1}{2}\left((n-p)\log(2\pi\sigma^2) + \log\det(H) + \log\det(X^T H^{-1} X) + \frac{(y - X\hat{\beta})^T H^{-1}(y - X\hat{\beta})}{\sigma^2}\right) \\ &= -\frac{1}{2}\left((n-p)\log(2\pi\sigma^2) + \log\det(H) + \log\det(X^T H^{-1} X) + \frac{y^T P(H) y}{\sigma^2}\right), \qquad (6.14) \end{aligned}$$

where $H = ZGZ^T + I$ and

$$P(H) = H^{-1} - H^{-1}X(X^T H^{-1} X)^{-1} X^T H^{-1}$$

as in (6.12). For a derivation of the residual log-likelihood (6.14) we refer to [40, Section 2.2.5] and [111, Section 2.2.5]. In practice, the residual log-likelihood function (6.14) is often to be preferred due to the aforementioned biasedness of the maximum likelihood objective (6.13). For a thorough comparison of the two approaches, we refer to [56].

One can show that both the log-likelihood function (6.13) and the REML log-likelihood function (6.14) are bounded from above and a maximum likelihood estimator exists under suitable conditions [40, Theorem 4]. More precisely, existence is ensured if the rank of the combined design matrices of fixed and random effects is less than the number of observations n [40, Section 2.17].

To find a maximizer of the log-likelihood or the residual log-likelihood, earlier works propose to use Newton-type methods [87] or the Expectation Maximization (EM) algorithm [42]. For this, a parameterization of the random effects covariance matrix $G = G(\gamma)$, $\gamma \in \mathbb{R}^r$ and $r \ll q(q+1)/2$ is used as outlined in Section 6.1, and the log-likelihood or the residual log-likelihood is optimized with respect to γ. For the EM algorithm, we consider the random effects b as hidden information and alternate between updating the conditional expectation of $b|y$ and the variance parameters σ^2, G [87]. However, the slow convergence of EM is a well-known drawback for linear mixed models [56], [40, Section 2.12]. On the other hand, when starting too far from a local optimum, positive definiteness of the Hessian might not be given in Newton's method and convergence is not ensured [56]. A modification of Newton's method for linear mixed model is the Fisher scoring algorithm, where the Hessian is replaced by the negative expected information matrix in the Newton equation [56]. In case the linear mixed model is well-defined, the expected information matrix is positive definite, for details see [40, Section 2.11]. Yet, the Fisher scoring algorithm is computationally expensive [56]. For any optimizer we need to ensure that the random effects covariance matrix G is positive definite to have well-definedness in the functions (6.13) and (6.14). Typical approaches to ensure this consist in perturbing a singular iterate G^t by an adjustment matrix such that we get positive definiteness [40, Section 2.15.3] or to use the reparameterization $G = LL^T$, where L is a lower triangular matrix, via a Cholesky decomposition [40, Section 2.15.4]. The latter approach is used in the prominent `lme4` package [15] implemented in R for minimizing the *deviance* or the *profiled REML* criterion. A main feature of the `lme4` package compared to other packages like the `nlme` package [112, 111] is that it has an efficient implementation for crossed random effects which are usually harder to fit [15]. We explain the optimization approach implemented in the `lme4` package in the following.

6.2.3 Estimation of Variance Parameters with Profiled Log-likelihood and Profiled Residual Log-likelihood

We maximize the *profiled log-likelihood* or the *profiled residual log-likelihood* to get estimates for the variance parameters [56, 15], [40, Section 2.2.4]. For this, we fix the log-likelihood l_{ML} or the residual log-likelihood l_R at H and consider it as a function of σ^2, only. Since the functions l_{ML}, l_R are concave in σ^2, we get the maximizers

$$\hat{\sigma}^2_{ML} = \frac{y^T P(H) y}{n}, \tag{6.15}$$

$$\hat{\sigma}^2_{R} = \frac{y^T P(H) y}{n-p} \tag{6.16}$$

for the log-likelihood l_{ML} (6.13) and the residual log-likelihood l_R (6.14), respectively [56]. Plugging the expressions (6.15), (6.16) into minus twice the log-likelihood l_{ML} and minus twice the residual log-likelihood l_R, we get the *profiled deviance* $\tilde{l}_{ML}$ and the *profiled REML criterion* $\tilde{l}_R$, respectively, that is

$$\tilde{l}_{ML}(G) = \log\det(H) + n\left(1 + \log\left(\frac{2\pi y^T P(H) y}{n}\right)\right), \tag{6.17}$$

$$\tilde{l}_{R}(G) = \log\det(H) + \log\det(X^T H^{-1} X) + (n-p)\left(1 + \log\left(\frac{2\pi y^T P(H) y}{n-p}\right)\right), \tag{6.18}$$

see [15] and [40, Section 2.2.3 & 2.2.4]. Here, $H = I + ZGZ^T$ as in (6.6) and $P(H)$ as in (6.12). As $\tilde{l}_{ML}$, $\tilde{l}_R$ are independent of σ^2, we can minimize the respective function with respect to G, only.

The optimization approach implemented in the `lme4` package [15] uses the profiled deviance $\tilde{l}_{ML}$ or the profiled REML criterion $\tilde{l}_R$ as an objective function. The approach consists of a reparameterization of the covariance matrix $G \in \mathbb{R}^{q \times q}$ based on the block diagonal structure specified in (6.7). To that end, for each of the matrices $\Psi_j \in \mathbb{R}^{q_j \times q_j}$, a Cholesky decomposition is considered, that is

$$\Psi_j = V_j V_j^T, \tag{6.19}$$

where $V_j \in \mathbb{R}^{q_j \times q_j}$ is a lower triangular matrix. Then, a vectorization of the elements of the *template matrix* V_j is performed, that is we flatten the template matrix V_j and get a vector $\gamma_j \in \mathbb{R}^{q_j(q_j+1)/2}$. Thus, the matrix G can be fully characterized by the parameter $\gamma = (\gamma_1, \ldots, \gamma_K)$. This means that the approach in the `lme4` package [15] switches from finding a matrix $G \in \mathbb{R}^{q \times q}$ to a vector $\gamma \in \mathbb{R}^r$ of dimension $r = \sum_{j=1}^{K} q_j(q_j + 1)/2$. Considering an optimization problem with the vector-valued parameter γ, precaution is required that the reconstruction of Ψ_j via (6.19) yields a positive definite matrix. In the `lme4` package, this is realized by adding a lower box constraint for the diagonal elements of the template matrices V_j. Let γ_{j_i}, $i = 1, \ldots, m$ with $m \leq r$ be the diagonal elements of the template matrix V_j. Then we introduce the constraints

$$e_{low} \leq \gamma_{j_i}, \tag{6.20}$$

where $e_{low} > 0$ for each of the diagonal elements γ_{j_i} of the template matrix V_j. Optimizing with respect to parameters γ_j such that the diagonal elements γ_{j_i} of the template matrix V_j are positive ensures that the resulting covariance matrix Ψ_j is positive definite [105, Appendix A.2]. Typically, e_{low} is set to a small value, e.g. $e_{low} = 10^{-4}$ [44].

This approach allows for using any "general-purpose nonlinear optimizer" [15] that can handle box constraints. The authors of the `lme4` package suggest to use Powell's BOBYQA algorithm [114] or the Nelder Mead simplex algorithm [15]. The acronym BOBYQA of the former algorithm stands for *bound-optimization by quadratic optimization* [114]. It follows a trust-region scheme, where a quadratic approximation is used in every iteration which coincides with the objective function at a suitable number of interpolation points. The algorithm does not require the gradient or the Hessian of the objective and is thus a derivative-free method [114, 44]. As such, the accuracy of the method highly depends on the choice of interpolation points which is a hyperparameter in the method [114]. If the interpolation points are not suitably chosen, the quadratic model yields a poor approximation of the objective and the algorithm possibly ends up in slow convergence. The other default optimizer in the `lme4` package is the Nelder Mead method [50]. It is a heuristic search method which is based on polytopes of prespecified dimension and is also a derivative-free method, where box-constraints can easily be incorporated [113, 114]. However, a drawback of the Nelder Mead method is that it can converge to non-stationary points [113].

With the approach implemented in the `lme4` package, we might get a *singular fit* where we hit the boundary of the feasible parameter space and G is close to a singular matrix. This frequently occurs in practice for complex covariance structures (e.g. multiple correlated random slopes) and small to medium-sized data sets [15]. Although the `lme4` package allows for singular fits, they are usually not desirable as the chances of numerical problems get higher and the optimizer of choice possibly does not converge [44]. Besides, post-hoc inferential procedures may be inappropriate for singular fits [44].

Summing up, the `lme4` approach *profiles* the scaling parameter σ^2 out of the *REML* log-likelihood. For the optimization with respect to σ^2, the closed-form expressions (6.15), (6.16) are used. For the optimization with respect to the matrix G, we use a Cholesky decomposition and flatten the *template matrices* V_j, resulting in the parameter vector γ over which we iterate by using optimizers imposing box-constraints [15]. The use of aforementioned heuristic search methods to optimize with respect to γ makes convergence theory hard to study and does not use problem-specific gradient or Hessian information [50]. Besides, the approach of flattening the template matrices V_j neglects the matrix structure of G. Further, G must be positive definite which can be ensured by a suitable choice of the box constraints in `lme4` [44] but the lower parameter e_{low} is a hyperparameter and needs to be set in advance. Another issue with the approach is the occurence of singular fits which might yield numerical problems in practice.

Motivated by these considerations, we formulate the problem of finding optimal variance parameters σ^2, G as a Riemannian optimization problem in the following. As such, we take into account the curvature and the matrix structure of G and optimize for the residual variance σ^2 and the random effects covariance matrix G together.

6.3 Riemannian Approach for Variance Parameter Estimation

We make use of the Riemannian geometry of covariance matrices to exploit the geometric structure of the random effects covariance matrix G. To formulate the optimization problem as a Riemannian optimization problem, we take a closer look at the structure of the matrix H given by (6.6). We use the block-diagonal structure of the matrix G given by (6.7) and consider the corresponding design matrix Z. As outlined in Section 6.1, the matrix Z is composed of K blocks corresponding to the grouping variables, that is

$$Z = \left(Z^{(1)}, \dots, Z^{(K)}\right),$$

where $Z^{(j)} \in \mathbb{R}^{n \times \tilde{q}_j}$, $\tilde{q}_j = M_j q_j$. Following the construction of Z implemented by the `lme4` package [44], each of the grouping variable specific random effects design matrices $Z^{(j)}$ consists itself of M_j blocks $Z^{(j,l_j)}$, where each block $Z^{(j,l_j)}$ is of size $n \times q_j$, that is

$$Z^{(j)} = \left(Z^{(j,1)}, \dots, Z^{(j,M_j)}\right). \tag{6.21}$$

Here, the i-th row, $i = 1, \dots, n$, of $Z^{(j)}$ denoted by $\left(Z^{(j)}\right)_i$ has possibly nonzero elements in columns $(l_j - 1)q_j + 1, \dots, l_j q_j$ if and only if observation i arises from the l_j-th level of grouping factor $\mathcal{B}_j$, $l_j = 1, \dots, M_j$. Thus, the i-th row of the random effects design matrix Z has

$$\sum_{j=1}^{K} (M_j - 1)\, q_j \tag{6.22}$$

structural zeros. This means that Z is not fully populated and the sparsity increases with the number of levels M_j [15]. We exploit the specific structure of Z given by the expression (6.21) and rewrite the matrix H as

$$H = I + ZGZ^T = I + \sum_{j=1}^{K} Z^{(j)} G_j {Z^{(j)}}^T = I + \sum_{j=1}^{K} \sum_{l_j=1}^{M_j} Z^{(j,l_j)} \Psi_j {Z^{(j,l_j)}}^T, \tag{6.23}$$

where we used the block-diagonal structure of the random effects covariance matrix G given by (6.7). With the reformulation (6.23), we can express the function H as a function of the matrices Ψ_j. This allows us to consider the manifold of positive definite matrices of dimensions $\mathbb{P}^{q_j}$ induced by the Ψ_j instead of considering the manifold of positive definite matrices $\mathbb{P}^q$ induced by G. Typically, the dimension q_j of Ψ_j is very small (e.g. $q_j = 2$ for a grouping factor with a random intercept and one random slope), whereas the dimension of G can be very large if the number of levels M_j in the data set is large.

We formalize the Riemannian optimization problem for variance parameter estimation of the REML in the following.

6.3.1 Riemannian Setting, Gradient and Hessian

We consider the residual log-likelihood as presented in (6.14). The parameters of interest are the residual variance σ^2 and the random effects covariance matrix G which can be expressed via the matrix Ψ_j of lower dimension. For the residual variance, we introduce a variable $\eta \in \mathbb{R}$ and set

$$\eta = \log(\sigma^2)$$

similar to the Riemannian formulation for Gaussian mixture models in Chapter 5. Further, we use the derived relationship (6.23) between the covariance matrix H and the matrices Ψ_j and rewrite the REML log-likelihood (6.14) as

$$l_R(\theta) = -\frac{1}{2}\left((n-p)\log(2\pi) + (n-p)\eta + \log\det(H) + \log\det(X^T H^{-1} X) + \frac{y^T P(H) y}{\exp(\eta)}\right), \tag{6.24}$$

where $\theta = (\eta, \Psi)$ for $\Psi = (\Psi_1, \ldots, \Psi_K)$ and $H = I + \sum_{j=1}^{K} \sum_{l_j=1}^{M_j} Z^{(j,l_j)} \Psi_j Z^{(j,l_j)^T}$.
The expression $P(H)$ is given by (6.12). Since the random effects covariance matrix G is assumed to be positive definite, we must require that the matrices $\Psi_j \in \mathbb{R}^{q_j \times q_j}$ are positive definite. Thus, maximizing the REML objective (6.24) is a Riemannian optimization problem over the manifold

$$\mathcal{M}_{LME} = \mathbb{R} \times \left(\bigtimes_{j=1}^{K} \mathbb{P}^{q_j} \right), \tag{6.25}$$

where $\mathbb{P}^{q_j}$ is the manifold of positive definite matrices of dimension q_j as introduced in Chapter 4. We summarize the problem in the following.

Problem:

Let $\theta = (\eta, \Psi)$ with $\Psi = (\Psi_1, \dots, \Psi_K)$, $\Psi_j \succ 0$, $\Psi_j \in \mathbb{R}^{q_j \times q_j}$. For given $X \in \mathbb{R}^{n\times p}$, $Z = (Z^{(1)}, \dots, Z^{(K)}) \in \mathbb{R}^{n\times q}$ with $Z^{(j)} = (Z^{(j,1)}, \dots, Z^{(j,M_j)})$, $Z^{(j,l_j)} \in \mathbb{R}^{n\times q_j}$ and $y \in \mathbb{R}^n$, we consider the Riemannian optimization problem

$$\min_{\theta\in\mathcal{M}_{LME}} L_R(\theta) = (n-p)\eta + \log\det(H) + \log\det(X^T H^{-1} X) + \frac{y^T P(H) y}{\exp(\eta)}, \qquad (6.26)$$

where

$$H = I + \sum_{j=1}^{K} \sum_{l_j=1}^{M_j} Z^{(j,l_j)} \Psi_j {Z^{(j,l_j)}}^T, \quad P(H) \ = H^{-1} - H^{-1}X(X^T H^{-1} X)^{-1} X^T H^{-1}$$

and

$$\mathcal{M}_{LME} = \mathbb{R} \times \left(\bigtimes_{j=1}^{K} \mathbb{P}^{q_j} \right).$$

We note that we can formulate the maximization of the log-likelihood l_{ML} (6.13) as an optimization problem over the product manifold (6.25) accordingly if we replace the fixed effects parameters β by its estimator given by (6.10). The resulting objective can be easily found by replacing the factor $n-p$ by n in (6.24) and by dropping the term $\log\det(X^T H^{-1} X)$. For this reason, we only consider REML estimation in the following as the consideration of ML is straightforward. Similar to the profiled approach in the `lme4` package [44], we minimize minus twice the function l_R in our approach, where l_R is as in (6.14). Further, we drop the term $(n-p)\log(2\pi)$ as it does not affect the optimization.

Riemannian setting for linear mixed models

The objective function (6.26) is a smooth function over the product manifold (6.25) consisting of the real scalars $\mathbb{R}$ and the K manifolds of positive definite matrices denoted by $\mathbb{P}^{q_j}$. Following the considerations of the manifold $\mathbb{P}^{q_j}$ in Chapter 4, we get the tangent space of the manifold (6.25) by

$$T_\theta \mathcal{M}_{LME} = \mathbb{R} \times \left(\bigtimes_{j=1}^{K} \mathbb{S}^{q_j} \right), \tag{6.27}$$

where $\mathbb{S}^{q_j}$ denotes the set of symmetric matrices of dimension q_j.

In compliance with the considerations about metrics for product manifolds on the manifold of positive definite matrices in Section 2.1.3 and Section 4.2.1, we define an inner product $\langle \cdot, \cdot \rangle_\theta$ on $T_\theta \mathcal{M}_{LME}$ by

$$\langle \xi, \chi \rangle_\theta = \xi_\eta \chi_\eta + \sum_{j=1}^{K} \operatorname{tr}(\Psi_j^{-1} \xi_{\Psi_j} \Psi_j^{-1} \chi_{\Psi_j}), \tag{6.28}$$

where $\theta = (\eta, \Psi) \in \mathcal{M}_{LME}$, $\Psi = (\Psi_1, \dots, \Psi_K)$, $\xi_\theta = (\xi_\eta, \xi_\Psi), \chi_\theta = (\chi_\eta, \chi_\Psi) \in T_\theta \mathcal{M}$ with $\xi_\Psi = (\xi_{\Psi_1}, \dots, \xi_{\Psi_K})$, $\chi_\Psi = (\chi_{\Psi_1}, \dots, \chi_{\Psi_K})$.

Accordingly, the retraction associated with the product manifold (6.25) and the inner product (6.28) reads

$$R_\theta(\xi_\theta) = \begin{pmatrix} \eta + \xi_\eta \\ \Psi_1 \exp\left(\Psi_1^{-1} \xi_{\Psi_1}\right) \\ \vdots \\ \Psi_K \exp\left(\Psi_K^{-1} \xi_{\Psi_K}\right) \end{pmatrix}, \tag{6.29}$$

where $\theta \in \mathcal{M}_{LME}$ and $\xi_\theta \in T_\theta \mathcal{M}_{LME}$ as above, see (4.16).

In the following, we derive expressions for the Riemannian gradient and the Riemannian Hessian of the problem (6.26). These contribute to a deeper understanding of the underlying geometry of the optimization problem and allow for higher-order Riemannian optimizers in order to solve (6.26).

Riemannian gradient for linear mixed models

We specify the Riemannian gradient for the problem (6.26) based on the inner product (6.28).

Theorem 6.2. *For $\theta \in \mathcal{M}_{LME}$, the Riemannian gradient* $\operatorname{grad} L_R(\theta) \in T_\theta \mathcal{M}_{LME}$ *of problem* (6.26) *is given by*

$$\operatorname{grad} L_R(\theta) = \begin{pmatrix} \chi_\eta \\ \chi_\Psi \end{pmatrix}, \tag{6.30}$$

where $\chi_\Psi = (\chi_{\Psi_1}, \dots, \chi_{\Psi_K})$ *and*

$$\chi_\eta = (n-p) - \frac{y^T P(H) y}{\exp(\eta)}, \qquad \chi_{\Psi_j} = \Psi_j \sum_{l_j=1}^{M_j} Z^{(j,l_j)^T} \operatorname{grad^e}_H L_R(\theta) Z^{(j,l_j)} \Psi_j$$

with

$$\operatorname{grad^e}_H L_R(\theta) = P(H) + \frac{1}{\exp(\eta)} \operatorname{grad^e}_H (y^T P(H) y), \tag{6.31}$$

where

$$\begin{aligned} \operatorname{grad^e}_H (y^T P(H) y) = \Big(&- H^{-1} y y^T H^{-1} + H^{-1} y v_1^T H^{-1} + H^{-1} v_1 y^T H^{-1} \\ &- H^{-1} X (X^T H^{-1} X^T)^{-1} X^T v_2 v_2^T X (X^T H^{-1} X^T)^{-1} X^T H^{-1} \Big), \end{aligned} \tag{6.32}$$

and $v_1 = X(X^T H^{-1} X)^{-1} X^T H^{-1} y$, $v_2 = H^{-1} y$, $P(H)$ *as in* (6.12).

Proof. In analogy to the proof of Theorem 5.6 for the Riemannian gradient for Gaussian mixture models, we can express the Riemannian gradient for linear mixed models as

$$\operatorname{grad} L_R(\theta) = \begin{pmatrix} \operatorname{grad}_\eta L_R(\theta) \\ \operatorname{grad}_{\Psi_1} L_R(\theta) \\ \vdots \\ \operatorname{grad}_{\Psi_K} L_R(\theta) \end{pmatrix},$$

where $\operatorname{grad}_\eta L_R(\theta) \in \mathbb{R}$, $\operatorname{grad}_{\Psi_j} L_R(\theta) \in \mathbb{S}^{q_j}$ denotes the gradient with regard to η and Ψ_j, respectively.

We first specify the Euclidean gradient $\operatorname{grad}_\eta L_R(\theta)$ of L_R with respect to η. Here, the Riemannian gradient is equal to the Euclidean gradient and reads

$$\operatorname{grad}_\eta L_R(\theta) = (n-p) - \frac{1}{\exp(\eta)} y^T P(H) y,$$

yielding the expression for χ_η.

The Riemannian gradient with respect to Ψ_j under the described setting is given by the projected Euclidean gradient onto the tangent space $\mathbb{S}^{q_j}$ equipped with the affine-invariant metric (4.12), that is

$$\operatorname{grad}_{\Psi_j} L_R(\theta) = \frac{1}{2} \Psi_j \left(\operatorname{grad^e}_{\Psi_j} L_R(\theta) + \left(\operatorname{grad^e}_{\Psi_j} L_R(\theta) \right)^T \right) \Psi_j, \tag{6.33}$$

where $\operatorname{grad^e}_{\Psi_j} L_R(\theta)$ denotes the Euclidean gradient of the Euclidean smooth extension of $L_R(\theta)$ with respect to Ψ_j. We now specify the Euclidean gradient $\operatorname{grad^e}_{\Psi_j}$ for all $j = 1, \dots, K$. By the chain rule, we get

$$\operatorname{grad^e}_{\Psi_j} L_R(\theta) = \sum_{l_j=1}^{M_j} Z^{(j,l_j)^T} \operatorname{grad^e}_H L_R(\theta) Z^{(j,l_j)}, \tag{6.34}$$

where $\operatorname{grad^e}_H L_R(\theta)$ is the Euclidean gradient of L_R with respect to the matrix $H \in \mathbb{R}^{n \times n}$ given by (6.23), that is

$$H = I + \sum_{j=1}^{K} \sum_{l_j=1}^{M_j} Z^{(j,l_j)} \Psi_j Z^{(j,l_j)^T}.$$

By applying the chain rule, we obtain

$$\operatorname{grad^e}_H L_R(\theta) = H^{-1} - H^{-1}X(X^T H^{-1} X^T)^{-1} X^T H^{-1} + \frac{1}{\exp(\eta)} \left(\operatorname{grad^e}_H (y^T P(H) y) \right). \tag{6.35}$$

We define

$$u_1(H) := y^T H^{-1} y, \qquad u_2(H) := y^T H^{-1} X (X^T H^{-1} X^T)^{-1} X^T H^{-1} y,$$

hence the last expression in (6.35) is given by

$$\operatorname{grad^e}_H (y^T P(H) y) = \operatorname{grad^e}(u_1(H)) - \operatorname{grad^e}(u_2(H)). \tag{6.36}$$

With the Leibniz rule, we get

$$\begin{aligned} \operatorname{grad^e}(u_1(H)) &= -H^{-1} y y^T H^{-1} \\ \operatorname{grad^e}(u_2(H)) &= -\Big(H^{-1} y v_1^T H^{-1} + H^{-1} v_1 y^T H^{-1} \\ &\qquad - H^{-1} X (X^T H^{-1} X^T)^{-1} X^T v_2 v_2^T X (X^T H^{-1} X^T)^{-1} X^T H^{-1} \Big), \end{aligned} \tag{6.37}$$

where we have set $v_1 = X(X^T H^{-1} X)^{-1} X^T H^{-1} y$ and $v_2 = H^{-1} y$.
We plug (6.37) into (6.36), (6.35), (6.34) and use the relationship of the Euclidean and Riemannian gradient given by (6.33). The symmetry of the Euclidean gradient $\operatorname{grad^e}_{\Psi_j}$ with respect to Ψ_j yields the expression for $\operatorname{grad}_{\Psi_j} L_R(\theta)$. □

As outlined in Chapter 2, Riemannian optimizers often benefit from second-order information about the objective. Thus, besides deriving the Riemannian gradient for linear mixed models in Theorem 6.2, we present an expression for the Riemannian Hessian of the objective (6.26) in the following.

Riemannian Hessian for linear mixed models

The following theorem states a formula for the Hessian of the REML objective for linear mixed models (6.26).

Theorem 6.3. *Let $\theta \in \mathcal{M}_{LME}$ and $\xi_\theta \in T_\theta \mathcal{M}_{LME}$, $\xi_\theta = (\xi_\eta, \xi_\Psi)$ with $\xi_\Psi = (\xi_{\Psi_1}, \dots, \xi_{\Psi_K})$. The Riemannian Hessian of problem* (6.26) *is given by*

$$\operatorname{Hess} L_R(\theta)[\xi_\theta] = \begin{pmatrix} \zeta_\eta \\ \zeta_\Psi \end{pmatrix} \in T_\theta \mathcal{M}_{LME},$$

where

$$\zeta_\eta = \frac{1}{\exp(\eta)} \Bigg(\xi_\eta y^T P(H) y - \sum_{j=1}^{K} y^T \Bigg(h_1 \left(\sum_{l_j=1}^{M_j} Z^{(j,l_j)} \xi_{\Psi_j} Z^{(j,l_j)^T} \right) + h_2 \left(\sum_{l_j=1}^{M_j} Z^{(j,l_j)} \xi_{\Psi_j} Z^{(j,l_j)^T} \right) \Bigg) y \Bigg),$$

and $\zeta_\Psi = (\zeta_{\Psi_1}, \dots, \zeta_{\Psi_K})$ with

$$\begin{aligned} \zeta_{\Psi_j} = \Psi_j \Bigg(& \sum_{r=1}^{K} \sum_{l_j=1}^{M_j} Z^{(j,l_j)^T} \mathrm{D}_H \left({\operatorname{grad}^{\mathrm{e}}}_H L_R(\theta) \right) [\sum_{l_r=1}^{M_r} Z^{(r,l_r)} \xi_{\Psi_r} Z^{(r,l_r)^T}] Z^{(j,l_j)} \\ & - \frac{\xi_\eta}{\exp(\eta)} \sum_{l_j=1}^{M_j} Z^{(j,l_j)^T} {\operatorname{grad}^{\mathrm{e}}}_H (y^T P(H) y) Z^{(j,l_j)} \Bigg) \Psi_j \\ & + \frac{1}{2} \Bigg(\xi_{\Psi_j} \sum_{l_j=1}^{M_j} Z^{(j,l_j)^T} {\operatorname{grad}^{\mathrm{e}}}_H L_R(\theta) Z^{(j,l_j)} \Psi_j \\ & + \Psi_j \sum_{l_j=1}^{M_j} Z^{(j,l_j)^T} {\operatorname{grad}^{\mathrm{e}}}_H L_R(\theta) Z^{(j,l_j)} \xi_{\Psi_j} \Bigg) \end{aligned}$$

Here,

$$\mathrm{D}_H \left({\operatorname{grad}^{\mathrm{e}}}_H L_R(\theta) \right) [\xi] = h_1(\xi) + h_2(\xi) + \frac{1}{\exp(\eta)} (h_3(\xi) - h_4(\xi)),$$

where

$$h_1(\xi) = -H^{-1} \xi H^{-1} \tag{6.38}$$

$$\begin{aligned} h_2(\xi) = {} & H^{-1} \xi H^{-1} X (X^T H^{-1} X)^{-1} X^T H^{-1} + H^{-1} X (X^T H^{-1} X)^{-1} X^T H^{-1} \xi H^{-1} \\ & - H^{-1} X (X^T H^{-1} X)^{-1} X^T H^{-1} \xi H^{-1} X (X^T H^{-1} X)^{-1} X^T H^{-1} \end{aligned} \tag{6.39}$$

$$h_3(\xi) = H^{-1}\xi H^{-1}yy^T H^{-1} + H^{-1}yy^T H^{-1}\xi H^{-1} \tag{6.40}$$

$$\begin{aligned} h_4(\xi) = &-(h_1(\xi)+h_2(\xi))\, yy^T H^{-1}X(X^T H^{-1}X)^{-1}H^{-1} + H^{-1}yy^T h_2(\xi) \\ &+ \Big(-(h_1(\xi)+h_2(\xi))\, yy^T H^{-1}X(X^T H^{-1}X)^{-1}H^{-1} + H^{-1}yy^T h_2(\xi)\Big)^T . \end{aligned} \tag{6.41}$$

The expressions $\operatorname{grad^e}_H L_R(\theta)$, $\operatorname{grad^e}_H(y^T P(H)y)$ *are given by* (6.31) *and* (6.32), *respectively.*

Proof. From the relationship (2.5), we get

$$\operatorname{Hess} L_R(\theta)[\xi_\theta] = \nabla_\theta \operatorname{grad} L_R(\theta) = \begin{pmatrix} \zeta_\eta \\ (\zeta_{\Psi_j})_{j=1,\dots,K} \end{pmatrix} = \begin{pmatrix} \nabla^e_{\xi_\eta} \operatorname{grad} L_R(\theta) \\ \left(\nabla^{pd}_{\xi_{\Psi_j}} \operatorname{grad} L_R(\theta)\right)_{j=1,\dots,K} \end{pmatrix} \tag{6.42}$$

where $\nabla^e_{\xi_\eta}$ denotes the classical Euclidean vector field differentiation along direction ξ_η and $\nabla^{pd}_{\xi_{\Psi_j}}$ denotes the Riemannian connection for positive definite matrices specified in (4.13).

For the Hessian at position η denoted by ζ_η, we observe that

$$\zeta_\eta = \nabla^e_{\xi_\eta} \operatorname{grad} L_R(\theta) = \mathrm{D}_\eta \left(\operatorname{grad^e}_\eta L_R(\theta)\right)[\xi_\eta] + \sum_{j=1}^{K} \mathrm{D}_{\Psi_j}\left(\operatorname{grad^e}_\eta L_R(\theta)\right)[\xi_{\Psi_j}].$$

We get

$$\mathrm{D}_\eta \left(\operatorname{grad^e}_\eta L_R(\theta)\right)[\xi_\eta] = \frac{\xi_\eta}{\exp(\eta)} y^T P(H) y,$$

and

$$\begin{aligned} \mathrm{D}_{\Psi_j}\left(\operatorname{grad^e}_\eta L_R(\theta)\right)[\xi_{\Psi_j}] &= -\frac{1}{\exp(\eta)} \mathrm{D}_{\Psi_j}(y^T P(H)y)[\xi_{\Psi_j}] \\ &= -\frac{1}{\exp(\eta)} \mathrm{D}_H(y^T P(H)y)[\sum_{l_j=1}^{M_j} Z^{(j,l_j)} \xi_{\Psi_j} Z^{(j,l_j)^T}]. \end{aligned} \tag{6.43}$$

By applying the chain rule on (6.43), we get the expression for ζ_η in Theorem 6.3.

For the Riemannian Hessian at position Ψ_j denoted by ζ_{Ψ_j}, using (2.5) and (4.14), we get

$$\begin{aligned} \zeta_{\Psi_j} &= \nabla^{pd}_{\xi_{\Psi_j}} \operatorname{grad} L_R(\theta) \\ &= \mathrm{D}_\theta(\operatorname{grad}_{\Psi_j} L_R(\theta))[\xi_\theta] - \frac{1}{2}\left(\xi_{\Psi_j} {\Psi_j}^{-1} \operatorname{grad}_{\Psi_j} L_R(\theta) + \operatorname{grad}_{\Psi_j} L_R(\theta) {\Psi_j}^{-1} \xi_{\Psi_j}\right) \end{aligned}$$

$$
\begin{aligned}
&= \mathrm{D}_\eta(\mathrm{grad}_{\Psi_j} L_R(\theta))[\xi_\eta] + \sum_{r=1}^{K} \mathrm{D}_{\Psi_r}\left(\mathrm{grad}_{\Psi_j} L_R(\theta)\right)[\xi_{\Psi_r}] \\
&\qquad - \frac{1}{2}\left(\xi_{\Psi_j}{\Psi_j}^{-1}\,\mathrm{grad}_{\Psi_j} L_R(\theta) + \mathrm{grad}_{\Psi_j} L_R(\theta){\Psi_j}^{-1}\xi_{\Psi_j}\right) \\
&= \mathrm{D}_\eta(\mathrm{grad}_{\Psi_j} L_R(\theta))[\xi_\eta] + \Psi_j\left(\sum_{r=1}^{K} \mathrm{D}_{\Psi_r}\left(\mathrm{grad^e}_{\Psi_j} L_R(\theta)\right)[\xi_{\Psi_r}]\right)\Psi_j \\
&\qquad + \frac{1}{2}\left(\xi_{\Psi_j}\,\mathrm{grad^e}_{\Psi_j} L_R(\theta)\Psi_j + \Psi_j\,\mathrm{grad^e}_{\Psi_j} L_R(\theta)\xi_{\Psi_j}\right),
\end{aligned}
\tag{6.44}
$$

where we used (6.34). For the first term in (6.44), we get

$$
\mathrm{D}_\eta(\mathrm{grad}_{\Psi_j} L_R(\theta))[\xi_\eta] = -\frac{\xi_\eta}{\exp(\eta)}\Psi_j\left(\sum_{l_j=1}^{M_j} Z^{(j,l_j)}\,\mathrm{grad^e}_H(y^T P(H)y){Z^{(j,l_j)}}^T\right)\Psi_j, \tag{6.45}
$$

where $\mathrm{grad^e}_H(y^T P(H)y)$ is given by (6.32).

For the second term in (6.44), we get

$$
\begin{aligned}
\mathrm{D}_{\Psi_r}\left(\mathrm{grad^e}_{\Psi_j} L_R(\theta)\right)[\xi_{\Psi_r}] &= \sum_{l_j=1}^{M_j} Z^{(j,l_j)}\,\mathrm{D}_{\Psi_r}\left(\mathrm{grad^e}_H L_R(\theta)\right)[\xi_{\Psi_r}]{Z^{(j,l_j)}}^T \\
&= \sum_{l_j=1}^{M_j} Z^{(j,l_j)}\,\mathrm{D}_H\left(\mathrm{grad^e}_H L_R(\theta)\right)[\sum_{l_r=1}^{M_r} Z^{(r,l_r)}\xi_{\Psi_r}{Z^{(r,l_r)}}^T]{Z^{(j,l_j)}}^T
\end{aligned}
\tag{6.46}
$$

by the chain rule. After applying the Leibniz rule on (6.46) several times and rearrangement of terms, we get

$$
\mathrm{D}_H\left(\mathrm{grad^e}_H L_R(\theta)\right)[\xi] = h_1(\xi) + h_2(\xi) + \frac{1}{\exp(\eta)}(h_3(\xi) - h_4(\xi)), \tag{6.47}
$$

where h_1, h_2 as in (6.38), (6.39) and h_3, h_4 as in (6.40), (6.41).

Plugging (6.47) into (6.46) and then (6.46), (6.45) into (6.44), we get the expression for ζ_{Ψ_j} in Theorem 6.3. □

We have derived the Riemannian gradient and Riemannian Hessian for the REML objective in (6.24). As mentioned at the beginning of this section, we can easily transfer the problem for the ML objective (6.13) by replacing the factor $n-p$ with n and omitting the additive terms in the Riemannian gradient and Hessian that are related to the term $\log\det\left(X^T H^{-1} X\right)$, see (6.13), (6.14). In this thesis, we focus on the parameter estimation with the REML objective due to the aforementioned underestimation of the residual error σ^2 occurring with using the ML objective (see Section 6.2.2).

With the Riemannian approach presented here, we formulate the objective as a function of both σ^2 and G, that is we estimate the residual variance and the random effects covariance matrix simultaneously. As explained in Section 6.2.3, the approach implemented in the `lme4` package expresses the objective as a function of the random effects covariance matrix, only, by profiling σ^2 out of the objective. The Riemannian approach presented in this chapter can be transferred into an according profiled setting. This can be achieved by considering the profiled REML log-likelihood (6.18) as the objective function and to optimize over the product manifold

$$\mathcal{M}_{LME}^{profiled} = \bigtimes_{j=1}^{K} \mathbb{P}^{q_j}.$$

6.3.2 Algorithmic Considerations

When considering the objective (6.26) for Riemannian optimization, we observe that we need to evaluate $H \in \mathbb{R}^{n\times n}$ in every iteration as well as terms involving its inverse H^{-1}. Since the number of observations n is usually large in applications, precaution is required that the matrix H as well as terms involving its inverse H^{-1} are implemented efficiently. The matrix H^t at iteration t, $t = 0, 1, 2, \dots$ reads

$$H^t = I + Z(G^t)Z^T = I + \sum_{j=1}^{K} Z^{(j)}(G_j)^t {Z^{(j)}}^T,$$

where

$$G_j^t = \begin{pmatrix} \Psi_j^t & & \\ & \ddots & \\ & & \Psi_j^t \end{pmatrix}$$

is the random effects covariance matrix of the j-th grouping factor at iteration t and $\Psi_j^t \in \mathbb{P}^{q_j}$ is the coordinate iterate at iteration t, see (6.7). Recall that the grouping factor specific design matrices $Z^{(j)}$ are sparse. For this reason, we store the matrices ${Z^{(j)}}^T \in \mathbb{R}^{M_j q_j \times n}$ in a *compressed sparse column (csc)* format [28]. Further, each column of ${Z^{(j)}}^T$ consists of $(M_j - 1)\, q_j$ structural zeros, see (6.22). Thus, by construction of the matrix G_j^t, the positions of structural zeros in $Z^{(j)}(G_j^t){Z^{(j)}}^T$ are the same for all iterations t. This means that only the potential nonzero elements in the matrix H need to be updated in every iteration which is exploited in the implementation used for the numerical results in Section 6.4.

Further, we need to compute factors involving the matrix inverse $(H^t)^{-1}$ in every iteration t. For this, we use the *CHOLMOD* approach [141, 28] which exploits the sparsity pattern of ZGZ^T for the Cholesky decomposition [15]. For details on sparse Cholesky decomposition, we refer to [28].

Besides exploiting the sparsity of the random effects covariance matrix, an efficient implementation of the Riemannian gradient and the Riemannian Hessian in Theorem 6.2 and Theorem 6.3, respectively, is desirable. The expressions derived contain the Euclidean gradient with respect to H, that is $\text{grad}^{\text{e}}{}_H\, L_R(\theta)$, and its directional derivative $\text{D}_H(\text{grad}^{\text{e}}{}_H\, L_R(\theta))[\xi]$ which are of size $n \times n$. However, the Riemannian gradient and Hessian with respect to Ψ_j is of size $q_j \times q_j$ with $q_j \ll n$ and we can implement the gradient and Hessian by considering matrix products of size $q_j \times n$ with size $n \times n$ instead of performing matrix products of size $n \times n$ with size $n \times n$. This consideration was used for the implementation.

In the following section, we show numerical results for the Riemannian Newton trust-region method and the Riemannian nonlinear conjugate gradient method for the problem (6.26).

6.4 Numerical Experiments

For the numerical investigation of the introduced Riemannian approach for linear mixed models, we studied two research questions. The first one consists of the investigation of Riemannian optimizers as an alternative to the approach in the `lme4` package [44]. Section 6.4.3 deals with this investigation, we compare the quality of the Riemannian optimizers with the ones from the `lme4` package on different data sets. The second research question addresses *singular fits* that sometimes occur in challenging applications with the `lme4` package [44] and possibly yield numerical problems, see Section 6.2.3 and [40, Section 2.15]. For this, we investigate whether we can improve estimates for the variance parameters σ^2, G by applying the proposed Riemannian approach after a singular fit has been detected with the `lme4` optimizers. This is explored in Section 6.4.4.

6.4.1 Simulation Design

To test the Riemannian approach for linear mixed models as presented in Section 6.3, we conducted numerical tests for simulated data. For this, we created data sets with 2 grouping variables $\mathcal{B}_1$, $\mathcal{B}_2$ in a crossed design with different random effects covariance structures. Further, we used a balanced design [40, Section 2.2.1], that is we assumed that all n observations are distributed equally among the M_1, M_2 levels of grouping factors $\mathcal{B}_1$, $\mathcal{B}_2$, respectively. To generate the data sets, we followed a similar approach as suggested in [39]. We present the simulation design in the following.

We created $n = 1000$ observations for our simulation design. For the fixed effects, we considered one continuous fixed effect and a fixed intercept, thus $p = 2$. The fixed effects parameter was set to $\beta = (1, 2)^T$ for all experiments.

For the variance parameters, we set the residual variance for all experiments equal to $\sigma^2 = 0.1$ and created $n = 1000$ realizations of a Gaussian distribution with zero mean and variance σ^2 to get realizations of the residual error ε, see (6.3). To incorporate the random effects in the created data sets, we created M_1, M_2 categorical variables reflecting the levels of the grouping variables $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. These were then used to build the random effects design matrix Z. The numerical tests were conducted for different structures in the grouping variable specific covariance matrices G_1, G_2 (6.7). We generated realizations of the random effects parameter $b = (b^{(1)}, b^{(2)})$ with a Gaussian distribution with zero mean and the covariance matrix $G = \operatorname{diag}(G_1, G_2)$ according to (6.4).

With these choices, we created the response vector $y \in \mathbb{R}^n$ according to the linear mixed model (6.3), that is

$$y = X\beta + Zb + \varepsilon.$$

We created multiple data sets reflecting the same linear mixed model structure with the described simulation approach. We recall that for testing different optimizers, only the variables y, X and Z are available and we do not know the fixed effects parameter β or the random effects parameter b. As outlined in Section 6.2, we can estimate β and b by Theorem 6.1 if G and σ^2 are known. In our numerical tests, we thus focused on the problem of finding the variance parameters G and σ^2.

6.4.2 Choice of Optimizers for Numerical Tests

Equipped with the Riemannian gradient and the Riemannian Hessian derived in Theorem 6.2 and Theorem 6.3, respectively, we can make use of Riemannian optimizers introduced in Chapter 2. Due to their fast local convergence close to an optimum, we applied a Riemannian Newton trust-region method (Algorithm 3) and a Riemannian nonlinear conjugate gradient method (Algorithm 5) for the problem of variance estimation in linear mixed models as presented in (6.26). For the latter, we used the toolbox `pymanopt` [136]. We used the default option implemented in `pymanopt` for the step-length rule (α^t in Algorithm 5) by Hestenes and Stiefel [61], for an overview of different update rules for the Riemannian nonlinear conjugate gradient method we refer to [120, Section 4.2] and [105, Section 5.2]. For the Riemannian Newton trust-region algorithm, we used a tuned version independent from the toolbox `pymanopt` where we reused terms that occur in the objective as well as the Riemannian gradient and Hessian (see Section 5.3.2). For the quadratic subproblem in the Riemannian Newton trust-region algorithm (Algorithm 3), we used the truncated conjugate gradient method (Algorithm 4) as a solver. We did not use a preconditioner for the tCG method since we observed very small numbers of inner iterations in the quadratic subproblem during the conduction of the experiments. The initial trust-region radius was set to a default value at $\Delta_0 = 1$ and the hyperparameters of the R-NTR algorithm were set to the same values as for our problem of fitting Gaussian mixture models in Section 5.3.2.

As reviewed in Section 6.2.2, different optimization methods have been proposed in the past for fitting the variance parameters of linear mixed models. Due to the popularity of the `lme4` package [44], we compared the established Riemannian optimization approach for REML estimation with the approach implemented in the `lme4` package and presented in Section 6.2.3. For this, we used the default optimizers in the `lme4` package, the BOBYQA and the Nelder Mead method [15].

We initialized the random effects covariance matrices G_j^0 by the identity matrix and the residual variance $(\sigma^2)^0$ by the expression (6.16), these are the default initializations in the `lme4` package [44]. We stopped all methods when either the number of iterations exceeded 1000, when the relative difference in the objective between two subsequent iterations fell below 10^{-5} (for R-NTR only if we did not reject the tentative direction returned by tCG, see Section 5.3.2) or when the step length fell below 10^{-7}. While the optimizers used in `lme4` [44] are derivative-free methods, the aforementioned Riemannian optimizers compute a gradient in each iteration. Thus, we added another stopping criterion for the Riemannian optimizers, that is when the Riemannian norm of the gradient, $\|\operatorname{grad} L_R(\theta)\|_\theta$, fell below some threshold close to 0 (specified in Section 6.4.3 and Section 6.4.4).

We used Python 3.8 for the experiments in this chapter and conducted all experiments on an Intel Xeon E3-1200 at 1.90 GHz with 8 cores and 16GB RAM.

In the following section, we address the first research question, that is whether the Riemannian approach can be considered as an alternative to the approach in `lme4` for standard data sets.

6.4.3 Riemannian Optimizers as an Alternative to the `lme4` Approach

We tested the Riemannian approach introduced in this thesis with respect to its appropriateness for fitting variance parameters σ^2, G of a linear mixed model. For this, we created 100 data sets with $K = 2$ grouping factors $\mathcal{B}_1$, $\mathcal{B}_2$ following the explanations in Section 6.4.1. We set the number of levels for the grouping factors equal to $M_1 = 15$ and $M_2 = 10$, respectively. The Riemannian optimizers were stopped when one of the aforementioned stopping criteria was fulfilled or when the Riemannian norm of the gradient, $\|\operatorname{grad} L_R(\theta)\|_\theta$, fell below 10^{-3}. To address the first research question dealing with the appropriateness of the Riemannian approach, two settings were tested. We present the according results in the following.

The first setting is a crossed random effects design with two random intercepts which results in the dimensions $q_1 = q_2 = 1$. Thus, the structure of the grouping effect specific random effects covariance matrix is given by

$$G_j = \begin{pmatrix} \tau_j^2 & & \\ & \ddots & \\ & & \tau_j^2 \end{pmatrix}, \tag{6.48}$$

where τ_j is the standard deviation of the j-th random effect, $j = 1, 2$ and $G_1 \in \mathbb{R}^{15\times 15}$, $G_2 \in \mathbb{R}^{10\times 10}$. The random effects variance parameters were chosen as $\tau_1 = 1.2$ and $\tau_2 = 0.9$ and the residual variance as $\sigma^2 = 0.1$. As outlined before, the values were initialized at $\tau_1^0 = \tau_2^0 = 1$. Table 6.1 shows a summary of the optimization results of 100 simulation runs, i.e. of 100 generated data sets following the specified distribution. We observe that the Riemannian optimizers (R-NTR, R-CG) together with the `lme4` approach with the Nelder Mead optimizer (NELDERMEAD) show the best results in terms of the objective value L_R although the deviation from the true objective (av. deviation L_R) is slightly higher than for the BOBYQA optimizer. In terms of the mean squared errors of the τ_1, τ_2 parameters (MSE τ_1, τ_2), we observe that both the Riemannian Newton trust-region algorithm as well as the Nelder Mead optimizer show comparable results, whereas the BOBYQA optimizer shows a higher error. In contrast, for the residual standard deviation σ, BOBYQA shows the lowest mean squared errors which are slightly below the mean squared error of the other methods. The Riemannian Newton trust-region algorithm converged in comparably few iterations, underlining the fast local convergence of the method. However, the Riemannian optimizers show much higher overall runtimes compared to the `lme4` approach with BOBYQA or Nelder Mead. This can be mainly led back to the high computational costs of evaluating the Riemannian gradient (and the Riemannian Hessian for R-NTR) in every iteration, whereas the BOBYQA and the Nelder Mead algorithms are derivative-free and have very low per-iteration costs for parameters of low dimensions (here: dimension 2) [15].

Table 6.1: Simulation results for two random intercepts with $\tau_1 = 1.2$, $\tau_2 = 0.9$, $\sigma^2 = 0.1$ and $M_1 = 15$, $M_2 = 10$.

	R-NTR	R-CG	BOBYQA	NELDERMEAD
av. number of iterations	12.01	55.17	61.43	36.16
av. runtime (s)	6.47	24.76	0.05	0.04
av. L_R	-1.5474	-1.5474	-1.4659	-1.5474
av. deviation L_R	2.2505	2.2505	2.0669	2.2505
MSE τ_1	0.0427	0.0427	0.2336	0.0434
MSE τ_2	0.0426	0.0426	0.1508	0.0425
MSE σ	0.0468	0.0468	0.0446	0.0468

The appropriateness of Riemannian optimizers for linear mixed models was tested on a second setting, where a random slope for the second grouping factor was added, i.e. $q_2 = 2$. The structure of the covariance matrix G_1 belonging to the first grouping factor $\mathcal{B}_1$ is the same as before and given by (6.48), whereas the structure of the covariance matrix belonging to the second grouping factor $\mathcal{B}_2$ is given by

$$G_2 = \begin{pmatrix} \tau_{2_1}^2 & \rho_2 \tau_{2_1} \tau_{2_2} & & & \\ \rho_2 \tau_{2_1} \tau_{2_2} & \tau_{2_2}^2 & & & \\ & & \ddots & & \\ & & & \tau_{2_1}^2 & \rho_2 \tau_{2_1} \tau_{2_2} \\ & & & \rho_2 \tau_{2_1} \tau_{2_2} & \tau_{2_2}^2 \end{pmatrix} \tag{6.49}$$

For the simulation, we set $\tau_1 = \tau_{2_1} = \tau_{2_2} = 1$ and $\rho_2 = 0.1$. We used the same residual variance as above, $\sigma^2 = 0.1$, to create 100 data sets for the simulation. The simulation results for this setting are summarized in Table 6.2.

Table 6.2: Simulation results for two random effects $b^{(1)} \in \mathbb{R}$, $b^{(2)} \in \mathbb{R}^2$ with $\tau_1 = \tau_{2_1} = \tau_{2_2} = 1$, $\rho_2 = 0.1$, $\sigma^2 = 0.1$ and $M_1 = 15$, $M_2 = 10$.

	R-NTR	R-CG	BOBYQA	NELDERMEAD
av. number of iterations	21.55	30.6	79.88	83.28
av. runtime (s)	11.79	14.38	0.06	0.06
av. L_R	-1.5378	-1.5381	-1.5305	-1.5393
av. deviation L_R	2.2922	2.2933	2.2711	2.2969
MSE τ_1	0.0336	0.0341	0.0847	0.0342
MSE τ_{2_1}	0.0542	0.0493	0.3093	0.3860
MSE τ_{2_2}	0.2669	0.3040	0.8990	0.6209
MSE ρ_2	0.000114	0.000056	0.009555	0.00862
MSE σ	0.0466	0.0466	0.0464	0.0467

When comparing the mean squared errors for the random effects covariance matrix, we observe that the Riemannian optimizers show much better results which is visible for the second random effect in particular (MSE τ_{2_1}, τ_{2_2}). Those are remarkably low compared to the `lme4` optimizers. We also attain a much lower mean squared error for the correlation ρ_2 with the Riemannian optimizers. We observe that both Riemannian optimizers converge much faster than the `lme4` optimizers in terms of number of iterations whereas the runtimes are much higher which we also noticed for the first setting. The lowest value of L_R (i.e. highest REML log-likelihood) is attained by the `lme4` approach with the Nelder Mead optimizer followed by the Riemannian optimizers.

Summing up, the Riemannian optimizers can keep up with the `lme4` optimizers for both settings in terms of τ_1 for the intercept-only random effect. When considering an additional random slope as in the second setting, we noticed a much better model quality with the Riemannian optimizers which can be seen from the mean squared errors for the second grouping factor, i.e. MSE τ_{2_1}, MSE τ_{2_2}. A possible explanation is that the `lme4` approach vectorizes the parameters of the random effects covariance matrix, whereas the Riemannian approach takes the positive definiteness and the matrix structure of G_j into account.

6.4.4 Improving Singular Fits with Riemannian Optimization

A frequently arising issue with fitting complex linear mixed models is covariance singularity or *singular fits* as named in the `lme4` package [44], where we hit the boundary of the feasible space. As outlined in Section 6.2.3 and [15], [40, Section 2.15], this typically occurs if we have variance close to 0, a high correlation within $b^{(j)}$ or complex designs with many levels and few observations. Although the `lme4` package allows for singular fits, these are often unfavorable because they usually result in numerical problems (see Section 6.2.3). To address singular fits, we created data sets where at least one of the two default optimizers BOBYQA or Nelder Mead returned a singular fit with the `lme4` package.

The Riemannian approach presented in Section 6.3 uses the affine-invariant inner product for the manifold of positive definite matrices (see (4.6) and (6.28)) which potentially pulls iterates close to the boundary into the interior of the cone $\mathbb{P}^{q_j}$. Thus, intuitively, the Riemannian approach is deemed well-suited to improve singular fits. To investigate this numerically, we generated 30 data sets yielding a singular fit with `lme4` for two different settings. We then used the final iterates $(\sigma^2)^{t*}$, G^{t*} as a starting value for our Riemannian optimizers in order to investigate whether they can improve the fit. In case both BOBYQA and Nelder Mead threw a singular fit, the returned iterate of the optimizer with the higher REML log-likelihood l_R (i.e. lower objective L_R) was used. The Riemannian optimizers were stopped when either one of the stopping criteria from Section 6.4.2 was fulfilled or when the Riemannian norm of the gradient, $\|\operatorname{grad} L_R(\theta)\|_\theta$, fell below 10^{-4}.

For the first setting, data sets with 2 random effects each reflecting a random intercept were generated, that is $q_1 = q_2 = 1$ and the random effects covariance structure given by (6.48). $M_1 = 15$ and $M_2 = 10$ levels were created. According to this, data sets yielding a singular fit with `lme4` were generated by setting the random effects variance parameters very close to 0, i.e. $\tau_1 = 0.001$ and $\tau_2 = 10^{-8}$. The residual variance was set to $\sigma^2 = 0.1$. The results are summarized in Table 6.3.

We see that we have a slight improvement in the objective value for the Riemannian optimizers. In addition, the mean squared errors of τ_1 are improved for the Riemannian CG method, whereas the mean squared error for the standard deviation for the second grouping factor shows improved values with both of the Riemannian optimizers. Also the mean squared error of the residual variance σ shows a lower value when applying the Riemannian optimizers. Looking at the number of iterations, we observe that the

Table 6.3: Improving singular fits with Riemannian optimization: $\tau_1 = 0.001$, $\tau_2 = 10^{-8}$, $\sigma^2 = 0.1$ and $M_1 = 15$, $M_2 = 10$.

	R-NTR	R-CG	BOBYQA	NELDERMEAD
av. number of iterations (incl.singfit)	16.93	87.73	9.0	14.27
av. L_R	-1.74588	-1.74588	-1.745878	-1.745864
rel. improvement in L_R	-0.000008	-0.000008		
MSE τ_1	0.000255	0.000219	0.000545	0.000221
MSE τ_2	0.002497	0.002497	0.002577	0.002587
MSE σ	0.046706	0.046706	0.04673	0.046729

Riemannian Newton trust-region method converges in much less iterations compared to the Riemannian nonlinear CG method. During the execution of the experiments, CG steps were performed for solving the quadratic subproblem (see Algorithm 3) in R-NTR and only in very few cases the trust-region radius was exceeded or negative curvature was detected (Algorithm 4). This suggests that we have a good initialization with the `lme4` approach despite the singularity, resulting in a starting point close to a local optimum with positive definite Hessian [34, Chapter 7]. In such a case, the Riemannian Newton trust-region algorithm shows a fast local convergence (Theorem 2.32). In Figure 6.1, we show the plotted norm of the Riemannian gradient of the iterations for a representative generated data set. We observe that $\|\operatorname{grad} L_R(\theta^t)\|_{\theta^t}$ attains values close to zero much faster with the Riemannian Newton trust-region algorithm compared to the Riemannian nonlinear CG method, reflecting the aforementioned fast local convergence of R-NTR.

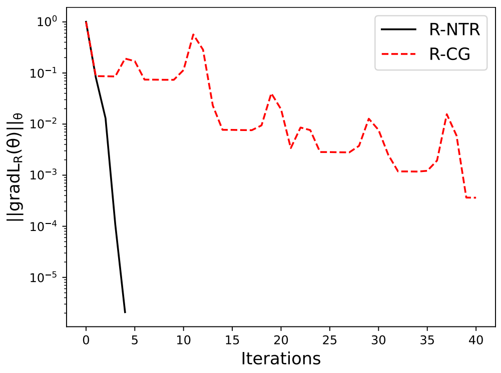


Figure 6.1: Norm of Riemannian gradient for improving `lme4` singular fits.

Another typical setting where the `lme4` approach frequently yields a singular fit is when we have many levels in the data set and only very few observations per level [40, Section 2.15]. To investigate this setting, the Riemannian methods were tested on data sets with more levels for the first grouping factor, that is $M_1 = 20$. Further, both a random intercept and a random slope was added for the first grouping factor, that is we set $q_1 = 2$ and the covariance matrix structure of the first grouping factor given by (6.49). Here, $\tau_{1_1} = \tau_{1_2} = 1$ and $\rho_1 = 0.1$ was chosen. For the second grouping variable, a random intercept model was imposed with covariance structure (6.48) and $\tau_2 = 1$, $M_2 = 10$ levels. The residual variance was chosen as before, that is $\sigma^2 = 0.1$. The simulation results are summarized in Table 6.4.

Table 6.4: Improving singular fits with Riemannian optimization: $\tau_{1_1} = 1$, $\tau_{1_2} = 1$, $\tau_2 = 1$, $\rho_1 = 0.1$, $\sigma^2 = 0.1$ and $M_1 = 20$, $M_2 = 10$.

	R-NTR	R-CG	BOBYQA	NELDERMEAD
av. number of iterations (incl.singfit)	104.9	106.73	71.33	98.17
av. L_R	-1.511814	-1.512387	-1.50476	-1.512825
rel. improvement in L_R	-0.00167	-0.002243		
MSE τ_{1_1}	0.3627	0.3765	0.3571	0.3876
MSE τ_{1_2}	0.4879	0.4846	0.8069	0.6844
MSE τ_2	1.6239	1.6291	1.8743	2.3232
MSE ρ_1	0.025269	0.025281	0.013011	0.027183
MSE σ	0.046989	0.046979	0.046741	0.047005

Considering Table 6.4, we observe an improvement for the variances of the random effects, that is in τ_{1_1}, τ_{1_2} and τ_2, whereas the improvement is strongest for τ_{1_2}. The correlation ρ_1 related to the first grouping factor is closest to the true value for the BOBYQA method, the Riemannian optimizers show slightly improved mean squared errors for ρ_1 compared to the Nelder Mead method. With the Riemannian optimizers, we gained a slight improvement in the objective by applying the Riemannian optimizers, but Nelder Mead shows the lowest objective value (highest REML log-likelihood). As observed before, the mean squared error in the residual standard deviation σ is lowest for the Riemannian optimizers, again indicating that we benefit from the simultaneous optimization of σ^2 and G with the Riemannian optimizers compared to the profiled approach by the `lme4` package. The number of iterations for the R-NTR method and the R-CG method are comparable in Table 6.4. This can be mainly led back to the observation that the truncated CG method for the subproblem in Algorithm 3 often stopped either because negative curvature in the Hessian was detected or because the trust-region radius was exceeded (see Algorithm 4). Frequently, the resulting truncated steps resulted in a bad quadratic approximation such that many steps were rejected and the trust-region radius was reduced until the termination criterion of a very small step length was satisfied (see Section 2.2.4 and Section 6.4.2). Like this, the R-NTR algorithm often got stuck in ill-behaved regions and did not yield a remarkable progress in the objective. Ways to get out of these plateaus could be to use a different approach to solve the quadratic subproblem yielding better iterates than the Cauchy point [34, Chapter 7], to use different hyperparameters for rejection thresholds and the initial trust-region algorithm [34, Chapter 17] or another adaption scheme of the trust-region radius in every iteration [34, Chapter 6]. Another possibility could be to first use a line-search method to get closer to a local optimum and then to start the R-NTR algorithm in order to increase the chance of attaining local superlinear convergence.

Summing up, we investigated whether we can improve singular fits with Riemannian optimizers on two typical settings. We observed a relative improvement of the objective with the Riemannian optimizers as well as lower mean squared errors for the random effects variances. The numerical results shown in this section indicate that the Riemannian optimizers pull back the iterates from the boundary of the feasible space and yield improved estimates of the variance components. In future work, this should be studied in depth for different structures of the random effects covariance matrix G by increasing the number of grouping factors K, the number of levels as well as the number of dimensions q_j or by assuming a strong correlation. Such settings typically yield singular fits with `lme4` [44] and it requires further investigation whether the Riemannian optimizers can improve the fits with such complicated structures. A deeper investigation of cases where the R-NTR gets stuck in the aforementioned ill-behaved plateaus should also be considered in the future.

6.4.5 Summary of Numerical Results

We tested the introduced Riemannian approach from Section 6.3 numerically with respect to two research questions. First, we investigated whether Riemannian optimizers show comparable results for the variance parameters of REML log-likelihood estimation as the popular `lme4` package. We observed that we got comparable or improved results with respect to the variance parameters. However, the runtimes of the Riemannian optimizers were much higher compared to the `lme4` optimizers which can be led back to the high computational cost of evaluating the Riemannian gradient and the Riemannian Hessian. For the second question, we investigated whether we could improve estimates that resulted in a singular fit with `lme4`. For this, we observed a slight improvement with the Riemannian optimizers indicating that the Riemannian approach is an appropriate tool for addressing singular fits. This should be further investigated in future work, especially for complex covariance structures of the random effects.

In the conducted experiments, the random effects covariance matrices were initialized at the identity matrix of appropriate dimension. Different approaches for a starting value of the matrix G have been proposed in the past, e.g. initialization by performing one step of the Fisher scoring algorithm [40, Section 2.13.1]. Another approach is based on *MINQUE* [40, Section 2.13 & 3.10.3], [56], namely *minimum norm quadratic unbiased estimator*, where a distribution-free estimation is imposed. Further suggestions can be found in [78]. As the performance of Riemannian derivative-based optimizers (especially R-NTR) highly depends on the starting value θ^0, their performance is expected to be further improved if we use an appropriate initialization strategy for G.

6.5 Concluding Remarks and Future Research Directions

Linear mixed models are typically used to model linear relationships in data, where the observations are sampled from different groups or clusters. Given an observed data set, parameter fitting of a linear mixed model can be divided into the estimation of the fixed and random effects parameters (Section 6.2.1) and the estimation of the variance parameters (Section 6.2.2). In this chapter, we focused on the estimation of the variance parameters σ^2 and G. Approaches like the `lme4` approach [15, 44] profile out the residual variance σ^2 of the objective and use a vectorization of the Cholesky decomposition of G in order to estimate the random effects covariance matrix (Section 6.2.3). Such an approach neglects the matrix structure of G and sometimes yields singular fits possibly leading to problems in real-world applications (see Section 6.2.3). We introduced a novel Riemannian framework for the fitting of the variance parameters σ^2, G in Section 6.3 which takes into account the geometric structure of the covariance matrix G and allows the simultaneous estimation of σ^2 and G. Based on the Riemannian framework, we derived a Riemannian gradient and a Riemannian Hessian which are used to apply Riemannian optimization methods introduced in Chapter 2. The numerical tests in Section 6.4 show the appropriateness of a Riemannian approach to variance parameter estimation in linear mixed models both as an alternative to existing methods as well as to improve singular fits. In the experiments conducted, we used small sized random effects. It is believed that for higher dimensions, optimizers based on the proposed Riemannian approach benefit from exploiting the geometric structure of covariance matrices in higher dimensions as well as the simultaneous estimation of σ^2 and G.

The Riemannian approach for linear mixed models introduced in this chapter was investigated with respect to classical linear mixed models of the form (6.3). The following extensions are conceivable and give rise to further exploration:

- *Maximizing the log-likelihood* (6.13) *with Riemannian optimization.* In our numerical tests in Section 6.4, we focused on the maximization of the REML log-likelihood. In some applications, it is more favorable to use the classical ML approach (6.13), for example if likelihood ratio tests are performed [56]. The Riemannian framework for REML estimation presented in this thesis can be easily transferred to ML estimation as outlined in Section 6.3 by dropping the term $\log\det(X^T H^{-1} X)$ in the objective. It is to be investigated whether the consideration of the ML objective still yields good results with the Riemannian approach compared to other optimizers, e.g. the ones from the `lme4` package studied in this chapter.

- *Consideration of penalized objective functions.* Like it is common for the classical linear model (6.2), a penalty term can be added to the objective in linear mixed models. This can be either a penalization term for the fixed effects parameter [122] or a regularization term that pushes the random effects covariance matrix G away from the boundary, e.g. by imposing a Bayesian prior on G and considering the *a posterior maximum likelihood (REML likelihood)* [32]. It would be interesting to explore whether the `lme4` optimizers with a penalization of singular fits [32] yield similar results for the random effects covariance matrix G compared to our Riemannian approach (Section 6.4.4).

- *Incorporation of covariance structures for the residual vector ε.* The linear mixed model (6.3) shows considerable flexibility in modeling covariance structures of the random effects. For the residual vector ε, we assume a Gaussian distribution with zero mean and variance $\sigma^2 I$, that is we assume that the within-group errors are iid with constant variance σ^2. However, some real-world applications require the modeling of heteroscedastic and/or correlated within-group errors such that the model (6.3) might not be appropriate [111, Chapter 5]. We can extend the classical linear mixed model (6.3) by assuming the more general distribution $\varepsilon \sim \mathcal{N}(0, R)$ for a positive definite covariance matrix R. Thus, the matrix R can be considered a parameter on the manifold of positive definite matrices and we can formulate the joint variance estimation of R, G as an optimization problem over the product manifold $\mathcal{M}_{LME}^{R} = \mathbb{P}^n \times \left(\bigtimes_{j=1}^{K} \mathbb{P}^{q_j} \right)$. The dimension of R is determined by the number of observations n which is usually large. However, typically R has a special structure [111, Section 5.2 & 5.3] and the entries can be identified with a vector $\omega \in \mathbb{R}^l$ with $l \ll n(n+1)/2$ (see remark (ii) in Section 6.1). Thus, it might be helpful to study different variance and correlation structures of the matrix R in depth. The consideration of special structures possibly yields an optimization problem over smaller covariance matrices similar to the dimension reduction achieved for G by (6.23), see Section 6.3.

- *Generalized linear mixed models.* Linear mixed models of the form (6.3) assume a Gaussian marginal distribution of the response vector y. A typical extension to allow for other distributions of the response y (e.g. discrete distributions) is to model data by a *generalized linear mixed model.* Here, the relationship between the linear predictor $X\beta + Zb$ and the response is modeled by a *link function* $g : \mathbb{R}^n \to \mathbb{R}^n$, that is

$$y = g(Zb + X\beta) + \varepsilon, \tag{6.50}$$

see [14] and [40, Chapter 7]. Considering generalized linear mixed models of the form (6.50) introduces some nonlinearity via the link function g which makes the model even more complex. The nonlinearity in generalized linear mixed models frequently occurs in singular fits [44, 15] which might yield numerical problems as outlined in Section 6.2.3 for linear mixed models. A future research direction could thus be to generalize the Riemannian framework introduced in Section 6.3 to generalized linear mixed models and to numerically study the behavior of Riemannian optimizers for singular fits similar to Section 6.4.4.

CHAPTER 7

Conclusion

In this thesis, we considered parameter estimation for two advanced statistical models with clustered data by Riemannian optimization. For this, we first introduced relevant foundations of differential geometry and built the basis to formulate optimization problems defined on Riemannian manifolds. Then, we introduced maximum likelihood estimation being the basis for the objectives that govern two statistical models of interest, namely a Gaussian mixture model and a linear mixed model. Maximum likelihood estimation for the two studied models incorporates the optimization with respect to covariance matrices that are assumed to be positive definite. Therefore, we built the bridge between Riemannian optimization and maximum likelihood estimation by introducing the Riemannian geometry of positive definite matrices. In that way, maximum likelihood estimation tasks with covariance matrix parameters can be formulated as Riemannian optimization problems. This formed the basis for the last two chapters where we studied the statistical models of interest: the Gaussian mixture model and the linear mixed model.

In the context of Gaussian mixture models, we studied the Riemannian formulation of maximum likelihood estimation as introduced by the works of Hosseini and Sra in [65, 66]. For the objective of interest, we derived the Riemannian Hessian which contributes to the geometric understanding of the respective parameter space. Based on the established expressions for higher-order information, we proposed a Riemannian Newton trust-region algorithm to fit the parameters of a Gaussian mixture model. The theoretical considerations were complemented by numerical tests for both a clustering and a probability density approximation task. We found that the Riemannian Newton trust-region algorithm could beat the popular Expectation Maximization (EM) algorithm both in terms of runtime and in terms of number of iterations. We observed this behavior in settings where there was a high overlap of the K clusters and the number of observations was sufficiently high. The introduced Riemannian Newton trust-region algorithm is deemed a practicable alternative to the EM algorithm as it exhibited faster results while showing similar errors (MSE/RMISE) and objective values in the experiments investigated in this thesis. This gives rise to further investigations of algorithms based on the framework studied in this thesis, especially for larger data sets which we outlined in Section 5.5. The contoured future research directions include the exploitation of the structure of the derived Riemannian Hessian for a higher number of observations and the incorporation of independence structures for larger dimensions.

After considering Gaussian mixture models, we studied variance parameter estimation in the context of linear mixed models. We introduced the model formally and presented the approach implemented in the prominent `lme4` package [44] that is often used for fitting linear mixed models in practice. Motivated by the drawbacks of this approach, we established a Riemannian formulation of variance parameter estimation in linear mixed models being one of the novelties introduced by this thesis. Based on the Riemannian formulation, we derived the Riemannian gradient and the Riemannian Hessian which contribute to a deeper understanding of the geometry of the parameter space in linear mixed models and allow to use derivative-based Riemannian optimizers. On top, we tested the appropriateness of Riemannian optimization for linear mixed models numerically and gave a computational comparison to optimization with the `lme4` approach [15]. We observed that we could reach similar errors as the `lme4` approaches and in some settings they were lower, indicating that Riemannian optimization yields estimates closer to the true values. Further, we tested the Riemannian approach for cases where optimizers of the `lme4` package [44] hit the boundary of the feasible space. We addressed such possibly unfavorable fits with Riemannian optimizers in the sense that we improved the objective slightly. The presented promising computational results of Riemannian optimization motivate to study further, more complex data sets, with the herein introduced Riemannian framework for linear mixed models. In Section 6.5, we described various future research directions which extend the work on linear mixed models presented in this thesis. This includes common generalizations of the linear mixed model which usually yields even more complex optimization problems.

The listed future research directions for Gaussian mixture models and linear mixed models in Chapter 5 and Chapter 6, respectively, demonstrate the various possibilities to extend the work within this thesis for the individual statistical models. Another interesting future research path consists of studying Riemannian optimization for the combination of the two models, that is to consider *mixtures of linear mixed models* [95]. Mixtures of linear mixed models allow to cluster data with a mixture model for dependent variables and to incorporate covariate information into the clustering process [104, 95]. Such extended models are used for example in the area of bioinformatics [25, 104] and are typically solved by the Expectation Maximization algorithm [104]. In this thesis, we observed remarkable improvements with a Riemannian approach compared to other optimizers for the individual statistical models. Therefore, applying Riemannian optimization for parameter estimation in mixtures of linear mixed models would potentially reveal chances and limitations of a geometric approach for more complex models.

List of Figures

List of Tables

Bibliography

[1] P.-A. Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. *Found. Comput. Math.*, 7(3):303–330, 2007.

[2] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds.* Princeton University Press, Princeton, NJ, 2008.

[3] Charu C. Aggarwal and Chandan K. Reddy. *Data Clustering - Algorithms and Applications.* CRC Press, Boca Raton, Fla, 2018.

[4] Murray Aitkin and Irit Aitkin. A hybrid EM/Gauss-Newton algorithm for maximum likelihood in mixture distributions. *Statistics and Computing*, 6(2):127–130, 1996.

[5] Awad H. Al-Mohy and Nicholas J. Higham. A New Scaling and Squaring Algorithm for the Matrix Exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2010.

[6] Marco Alfò, Luciano Nieddu, and Donatella Vicari. A Finite Mixture Model for Image Segmentation. *Statistics and Computing*, 18(2):137–150.

[7] Jason Altschuler, Sinho Chewi, Patrik Robert Gerber, and Austin J Stromme. Averaging on the Bures-Wasserstein manifold: dimension-free convergence of gradient descent. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[8] Jeffrey L. Andrews. Addressing overfitting and underfitting in Gaussian model-based clustering. *Computational Statistics and Data Analysis*, 127:160–171, 2018.

[9] Marc Arnaudon, Frédéric Barbaresco, and Le Yang. Riemannian medians and means with applications to radar signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 7(4):595–604, 2013.

[10] Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices. *SIAM J. Matrix Anal. Appl.*, 29:328–347, 2006.

[11] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[12] Charlotte Articus and Jan Pablo Burgard. A Finite Mixture Fay Herriot-type model for estimating regional rental prices in Germany. Research Papers in Economics 2014-14, University of Trier, Department of Economics, 2014.

[13] Afonso S Bandeira, Nicolas Boumal, and Amit Singer. Tightness of the maximum likelihood semidefinite relaxation for angular synchronization. *Mathematical Programming*, 163(1):145–167, 2017.

[14] Douglas Bates. Computational methods for mixed models. *Vignette for lme4*, 2011.

[15] Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. Fitting Linear Mixed-Effects Models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015.

[16] Ronny Bergmann. Manopt.jl: Optimization on manifolds in Julia. *Journal of Open Source Software*, 7(70):3866, 2022.

[17] Rajendra Bhatia. *Positive Definite Matrices.* Princeton University Press, Princeton, NJ, USA, 2007.

[18] Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the Bures-Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37, 12 2017.

[19] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, Berlin, Heidelberg, 2006.

[20] R.K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaiciulis, and W. Wittek. Methods for multidimensional event classification: a case study using images from a

Cherenkov gamma-ray telescope. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 516(2):511–528, 2004.

[21] N. Boumal. *Optimization and estimation on manifolds.* PhD thesis, Université catholique de Louvain, 2014.

[22] Nicolas Boumal. An introduction to optimization on smooth manifolds. To appear with Cambridge University Press, Apr 2022.

[23] Nicolas Boumal, Bamdev Mishra, P.-A. Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(42):1455–1459, 2014.

[24] Samuel Burer, Renato DC Monteiro, and Yin Zhang. Solving a class of semidefinite programs via nonlinear programming. *Mathematical Programming*, 93(1):97–122, 2002.

[25] Gilles Celeux, Olivier Martin, and Christian Lavergne. Mixture of linear mixed models for clustering gene expression profiles from repeated microarray experiments. *Statistical Modelling*, 5(3):243–267, 2005.

[26] Zeineb Chebbi and Maher Moakher. Means of Hermitian positive-definite matrices based on the log-determinant α-divergence function. *Linear Algebra and its Applications*, 436(7):1872–1889, 2012.

[27] Jiahua Chen and Xianming Tan. Inference for Multivariate Normal Mixtures. *Journal of Multivariate Analysis*, 100:1367–1383, 2009.

[28] Yanqing Chen, Timothy A Davis, William W Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):1–14, 2008.

[29] Anoop Cherian and Suvrit Sra. Positive definite matrices: data representation and applications to computer vision. In *Algorithmic advances in Riemannian geometry and applications*, pages 93–114. Springer, 2016.

[30] Anoop Cherian, Suvrit Sra, Arindam Banerjee, and Nikolaos Papanikolopoulos. Efficient similarity search for covariance matrices via the Jensen-Bregman LogDet Divergence. In *2011 International Conference on Computer Vision*, pages 2399–2406, 2011.

[31] Stéphane Chrétien and Alfred O Hero. On EM algorithms and their proximal generalizations. *ESAIM: Probability and Statistics*, 12:308–326, 2008.

[32] Yeojin Chung, Sophia Rabe-Hesketh, Vincent Dorie, Andrew Gelman, and Jingchen Liu. A nondegenerate penalized likelihood estimator for variance parameters in multilevel models. *Psychometrika*, 78(4):685–709, 2013.

[33] Giovanni Compiani and Yuichi Kitamura. Using Mixtures in Econometric Models: A Brief Review and Some New Results. *The Econometrics Journal*, 19(3):C95–C127, 2016.

[34] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust Region Methods.* Society for Industrial and Applied Mathematics, Philadelphia, USA, 2000.

[35] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009. Smart Business Networks: Concepts and Empirical Evidence.

[36] Morton L. Curtis. *Matrix groups.* Springer, New York, USA, 1979.

[37] Sanjoy Dasgupta. Learning Mixtures of gaussians. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, FOCS '99, page 634, USA, 1999. IEEE Computer Society.

[38] Neil E Day. Estimating the components of a mixture of normal distributions. *Biometrika*, 56(3):463–474, 1969.

[39] Lisa M DeBruine and Dale J Barr. Understanding mixed-effects models through data simulation. *Advances in Methods and Practices in Psychological Science*, 4(1), 2021.

[40] Eugene Demidenko. *Mixed Models - Theory and Applications.* Wiley, New York, 2004.

[41] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

[42] Arthur P Dempster, Donald B Rubin, and Robert K Tsutakawa. Estimation in covariance components models. *Journal of the American Statistical Association*, 76(374):341–353, 1981.

[43] Harold Doran, Douglas Bates, Paul Bliese, and Maritza Dowling. Estimating the multilevel Rasch model: With the lme4 package. *Journal of Statistical software*, 20:1–18, 2007.

[44] Ben Bolker Steven Walker Rune Haubo Bojesen Christensen Henrik Singmann Bin Dai Fabian Scheipl Gabor Grothendieck Peter Green John Fox Alexander Bauer Pavel N. Krivitsky Douglas Bates, Martin Maechler. *lme4: Linear Mixed-Effects Models using 'Eigen' and S4.*

[45] Denis Dresvyanskiy, Tatiana Karaseva, Vitalii Makogin, Sergei Mitrofanov, Claudia Redenbach, and Evgeny Spodarev. Detecting anomalies in fibre systems using 3-dimensional image data. *Stat. Comput.*, 30(4):817–837, 2020.

[46] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2017.

[47] Edward J. Dudewicz. *Introduction to Statistics and Probability.* Holt, Rinehart and Winston, Philadelphia, 1970.

[48] William Feller. *An Introduction to Probability Theory and Its Applications, Volume 2.* Wiley, New York, 1991.

[49] Tetsuya Fujie and Masakazu Kojima. *Journal of Global Optimization*, 10(4):367–380, 1997.

[50] Fuchang Gao and Lixing Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51(1):259–277, 2012.

[51] Reza Godaz, Benyamin Ghojogh, Reshad Hosseini, Reza Monsefi, Fakhri Karray, and Mark Crowley. Vector Transport Free Riemannian LBFGS for Optimization on Symmetric Positive Definite Matrix Manifolds. In *ACML*, volume 157 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 2021.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, 2016.

[53] Nicholas Gould, Dominique Orban, Annick Sartenaer, and Phillipe Toint. Sensitivity of trust-region algorithms to their parameters. *4OR quarterly journal of the Belgian, French and Italian Operations Research Societies*, 3:227–241, 2005.

[54] Marcus Gross, Ulrich Rendtel, Timo Schmid, Sebastian Schmon, and Nikos Tzavidis. *Estimating the density of ethnic minorities and aged people in Berlin: multivariate kernel density estimation applied to sensitive georeferenced administrative data protected via measurement error*, volume 180. Wiley Online Library, 2017.

[55] Ralitza Gueorguieva and John H Krystal. Move over ANOVA: progress in analyzing repeated-measures data andits reflection in papers published in the archives of general psychiatry. *Archives of general psychiatry*, 61(3):310–317, 2004.

[56] F.N. Gumedze and T.T. Dunne. Parameter estimation and inference in the linear mixed model. *Linear Algebra and its Applications*, 435(8):1920–1944, 2011.

[57] Andi Han, Bamdev Mishra, Pratik Jawanpuria, and Junbin Gao. On Riemannian Optimization over Positive Definite Matrices with the Bures-Wasserstein Geometry, 2021.

[58] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, New York, 2009.

[59] Richard J Hathaway. A constrained EM algorithm for univariate normal mixtures. *Journal of Statistical Computation and Simulation*, 23(3):211–230, 1986.

[60] Charles R Henderson. Estimation of genetic parameters. In *Biometrics*, volume 6, pages 186–187. International Biometric Soc 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210, 1950.

[61] Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

[62] Inbal Horev, Florian Yger, and Masashi Sugiyama. Geometry-aware principal component analysis for symmetric positive definite matrices. *Machine Learning*, 106(4):493–522, 2016.

[63] S.C. Horng. Examples of sublinear convergence of the EM algorithm. In *Proceedings of the Statistical Computing Section, American Statistical Association*, pages 266–271, 1987.

[64] R. Hosseini and M. Mash'al. Mixest: An estimation toolbox for mixture models. *CoRR*, abs/1507.06065, 2015.

[65] Reshad Hosseini and Suvrit Sra. Matrix manifold optimization for Gaussian mixtures. *Advances in Neural Information Processing Systems*, 28, 2015.

[66] Reshad Hosseini and Suvrit Sra. An alternative to EM for Gaussian mixture models: batch and stochastic Riemannian optimization. *Mathematical programming*, 181(1):187–223, 2020.

[67] Xiaoqiang Hua, Yongqiang Cheng, Hongqiang Wang, Yuliang Qin, Yubo Li, and Wenpeng Zhang. Matrix CFAR detectors based on symmetrized Kullback–Leibler and total Kullback–Leibler divergences. *Digital Signal Processing*, 69:106–116, 2017.

[68] Wen Huang, P.-A. Absil, Kyle A. Gallivan, and Paul Hand. ROPTLIB: An Object-Oriented C++ Library for Optimization on Riemannian Manifolds. *ACM Trans. Math. Softw.*, 44(4), 2018.

[69] Zhiwu Huang, Ruiping Wang, Shiguang Shan, Xianqiu Li, and Xilin Chen. Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 720–729. JMLR.org, 2015.

[70] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[71] Mortaza Jamshidian and Robert I. Jennrich. Conjugate Gradient Acceleration of the EM Algorithm. *Journal of the American Statistical Association*, 88(421):221–228, 1993.

[72] Hicham Janati, Boris Muzellec, Gabriel Peyré, and Marco Cuturi. Entropic optimal transport between unbalanced Gaussian measures has a closed form. *Advances in neural information processing systems*, 33:10468–10479, 2020.

[73] B. Jeuris, R. Vandebril, and B. Vandereycken. A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *ETNA*, 39:379–402, 2012.

[74] Harry Joe. *Dependence Modeling with Copulas.* Chapman and Hall/CRC Press, 2014.

[75] Heysem Kaya, Pmar Tüfekci, and Fikret S Gürgen. Local and global learning methods for predicting power of a combined gas & steam turbine. In *Proceedings of the international conference on emerging trends in computer and electronics engineering ICETCEE*, pages 13–18, 2012.

[76] Heysem Kaya, Pinar Tüfekci, and Erdinç Uzun. Predicting CO and NOx emissions from gas turbines: novel data and a benchmark PEMS. *Turkish Journal of Electrical Engineering and Computer Science*, 27:4783 – 4796, 2019.

[77] Konrad Königsberger. *Analysis II.* Springer, Berlin, Heidelberg, 1993.

[78] Nan Laird, Nicholas Lange, and Daniel Stram. Maximum likelihood computations with repeated measures: application of the EM algorithm. *Journal of the American Statistical Association*, 82(397):97–105, 1987.

[79] Lucien Le Cam. Maximum likelihood: an introduction. *International Statistical Review/Revue Internationale de Statistique*, pages 153–171, 1990.

[80] John M. Lee. *Riemannian Manifolds - An Introduction to Curvature.* Springer Science & Business Media, Berlin Heidelberg, 2006.

[81] John M. Lee. Smooth manifolds. In *Introduction to Smooth Manifolds*, pages 1–31. Springer New York, 2013.

[82] Taeyoon Lee and Frank C Park. A geometric algorithm for robust multibody inertial parameter identification. *IEEE Robotics and Automation Letters*, 3(3):2455–2462, 2018.

[83] Andrew Lewandowski, Chuanhai Liu, and Scott Vander Wiel. Parameter Expansion and Efficient Inference. *Statistical Science*, 25(4), November 2010.

[84] Yan Li and Lei Li. A Novel Split and Merge EM Algorithm for Gaussian Mixture Model. In *2009 Fifth International Conference on Natural Computation*, volume 6, pages 479–483, 2009.

[85] Zhenhua Lin. Riemannian Geometry of Symmetric Positive Definite Matrices via Cholesky Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 40(4):1353–1370, 2019.

[86] James K Lindsey et al. *Parametric statistical inference.* Oxford University Press, 1996.

[87] Mary J Lindstrom and Douglas M Bates. Newton–Raphson and EM algorithms for linear mixed-effects models for repeated-measures data. *Journal of the American Statistical Association*, 83(404):1014–1022, 1988.

[88] Chuanhai Liu, Donald B. Rubin, and Ying Nian Wu. Parameter Expansion to Accelerate EM: The PX-EM Algorithm. *Biometrika*, 85(4):755–770, 1998.

[89] Shunlong Luo and Qiang Zhang. Informational distance on quantum-state space. *Phys. Rev. A*, 69:032106, 2004.

[90] Jinwen Ma and Jianfeng Liu. The BYY annealing learning algorithm for Gaussian mixture with automated model selection. *Pattern Recognition*, 40(7):2029–2037, 2007.

[91] David A. Magezi. Linear mixed-effects models for within-participant psychology experiments: an introductory tutorial and free, graphical user interface (LMMgui). *Frontiers in Psychology*, 6, 2015.

[92] Anirudha Majumdar, Georgina Hall, and Amir Ali Ahmadi. Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:331–360, 2020.

[93] Luigi Malagò, Luigi Montrucchio, and Giovanni Pistone. Wasserstein Riemannian geometry of Gaussian densities. *Information Geometry*, 1(2):137–179, November 2018.

[94] Geoffrey J. McLachlan and Thriyambakam Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, New York, 2007.

[95] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual review of statistics and its application*, 6:355–378, 2019.

[96] Mayank Meghwanshi, Pratik Jawanpuria, Anoop Kunchukuttan, Hiroyuki Kasai, and Bamdev Mishra. McTorch, a manifold optimization library for deep learning. *arXiv preprint, arXiv:1810.01811*, 2018.

[97] Peter W Michor, Dénes Petz, and Attila Andai. The curvature of the Bogoliubov-Kubo-Mori scalar product on matrices. *Infinite Dimensional Analysis, Quantum Probability and Related Topics*, 3(2):1–14, 2000.

[98] Maher Moakher. A Differential Geometric Approach to the Geometric Mean of Symmetric Positive-Definite Matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3):735–747, 2005.

[99] Maher Moakher and Mourad Zéraï. The Riemannian geometry of the space of positive-definite matrices and its application to the regularization of positive-definite matrix-valued data. *Journal of Mathematical Imaging and Vision*, 40(2):171–187, 2011.

[100] José Morales and Jorge Nocedal. Automatic Preconditioning by Limited Memory Quasi-Newton Updating. *SIAM Journal on Optimization*, 10:1079–1096, 06 2000.

[101] Kevin P. Murphy. *Machine Learning - A Probabilistic Perspective.* MIT Press, Cambridge, 2012.

[102] Iftekhar Naim and Daniel Gildea. Convergence of the EM Algorithm for Gaussian Mixtures with Unbalanced Mixing Coefficients. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 1427–1431, Madison, WI, USA, 2012.

[103] Dan Nettleton. Convergence properties of the EM algorithm in constrained parameter spaces. *Canadian Journal of Statistics*, 27(3):639–648, 1999.

[104] Shu-Kay Ng, Geoffrey J McLachlan, Kui Wang, L Ben-Tovim Jones, and S-W Ng. A mixture model with random-effects components for clustering correlated gene-expression profiles. *Bioinformatics*, 22(14):1745–1752, 2006.

[105] Jorge Nocedal and Stephen Wright. *Numerical Optimization.* Springer Science and Business Media, Berlin Heidelberg, 2006.

[106] Dirk Ormoneit and Volker Tresp. Averaging, maximum penalized likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates. *IEEE Transactions on Neural Networks*, 9(4):639–650, 1998.

[107] Pablo A. Parrilo and Sanjay Lall. Semidefinite programming relaxations and algebraic optimization in control. *European Journal of Control*, 9:2–3, 2003.

[108] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[109] Xavier Pennec. Manifold-valued image processing with SPD matrices. In Xavier Pennec, Stefan Sommer, and Tom Fletcher, editors, *Riemannian Geometric Statistics in Medical Image Analysis*, pages 75–134. Academic Press, 2020.

[110] Franz Pernkopf and Djamel Bouchaffra. Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1344–1348, 2005.

[111] José Pinheiro and Douglas Bates. *Mixed-effects models in S and S-PLUS*. Springer Science & Business media, 2006.

[112] José Pinheiro, Douglas Bates, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2022. R package version 3.1-158.

[113] Michael JD Powell. On search directions for minimization algorithms. *Mathematical programming*, 4(1):193–201, 1973.

[114] Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, 26, 2009.

[115] Chunhong Qi, Kyle A. Gallivan, and P.-A. Absil. Riemannian bfgs algorithm with applications. In *Recent Advances in Optimization and its Applications in Engineering*, pages 183–192, Berlin, Heidelberg, 2010. Springer.

[116] Richard A Redner and Homer F Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM review*, 26(2):195–239, 1984.

[117] Ruslan Salakhutdinov, Sam T Roweis, and Zoubin Ghahramani. Optimization with EM and expectation-conjugate-gradient. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 672–679, 2003.

[118] Adam Santos, Eloi Figueiredo, Moisés Silva, Reginaldo Santos, Claudomiro Sales, and João CWA Costa. Genetic-based EM algorithm to improve the robustness of Gaussian mixture models for damage detection in bridges. *Structural Control and Health Monitoring*, 24(3):e1886, 2017.

[119] A. Sartenaer. Automatic determination of an initial trust region in nonlinear programming. *SIAM J. Sci. Comput.*, 18(6):1788–1803, November 1997.

[120] Hiroyuki Sato. *Riemannian Optimization and Its Applications.* SpringerBriefs in Electrical and Computer Engineering, 2021.

[121] Daniel R. A. Schallmo and Christopher A. Williams. *Digital Transformation Now! - Guiding the Successful Digitalization of Your Business Model.* Springer, Berlin, Heidelberg, 2018.

[122] Jürg Schelldorfer, Peter Bühlmann, and SARA VAN DE GEER. Estimation for high-dimensional linear mixed-effects models using L1-penalization. *Scandinavian Journal of Statistics*, 38(2):197–214, 2011.

[123] D.W. Scott. *Multivariate Density Estimation. Theory, Practice, and Visualization.* Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, New York, London, Sydney, 1992.

[124] Lena Sembach, Jan Pablo Burgard, and Volker Schulz. A Riemannian Newton trust-region method for fitting Gaussian mixture models. *Statistics and Computing*, 32(1), 2021.

[125] Hichem Snoussi and Ali Mohammad-Djafari. Penalized maximum likelihood for multivariate Gaussian mixture. In *AIP Conference proceedings*, volume 617, pages 36–46. American Institute of Physics, 2002.

[126] Peter Sondergaard. "Information is the oil of the 21st century, and analytics is the combustion engine". *Gartner Symposium/ITxpo 2011, October 16–20, Orlando*, 2011.

[127] Dominique Spehner and Miguel Orszag. Geometric quantum discord with Bures distance. *New Journal of Physics*, 15(10):103001, 2013.

[128] Vladimir Spokoiny and Thorsten Dickhaus. *Basics of Modern Mathematical Statistics.* Springer Berlin Heidelberg, Wiesbaden, 2014.

[129] Suvrit Sra. A new metric on the manifold of kernel matrices with application to matrix geometric means. volume 25, 2012.

[130] Suvrit Sra and Reshad Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739, 2015.

[131] Reinhold Stahl and Patricia Staab. Die Vermessung des Datenuniversums. pages 65–66. Springer Berlin Heidelberg, 2017.

[132] Trond Steihaug. The Conjugate Gradient Method and Trust Regions in Large Scale Optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.

[133] Rolf Sundberg. An iterative method for solution of the likelihood equations for incomplete data from exponential families. *Communication in Statistics-Simulation and Computation*, 5(1):55–64, 1976.

[134] Yann Thanwerdas and Xavier Pennec. Exploration of Balanced Metrics on Symmetric Positive Definite Matrices. In *International Conference on Geometric Science of Information*, pages 484–493. Springer, 2019.

[135] Yann Thanwerdas and Xavier Pennec. O(n)-invariant Riemannian metrics on SPD matrices. *arXiv preprint, arXiv:2109.05768*, 2021.

[136] James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17(137):1–5, 2016.

[137] Jesse van Oostrum. Bures-wasserstein geometry. *arXiv preprint, arXiv:2001.08056*, 2020.

[138] Rob van Tulder, Alain Verbeke, and Lucia Piscitello. Introduction: international business in the information and digital age–an overview of themes and challenges. *International Business in the Information and Digital Age*, 2018.

[139] Robert J Vanderbei and H Yurttan Benson. On formulating semidefinite programming problems as smooth convex nonlinear optimization problems. *Technical Report*, 2000.

[140] Bart Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[141] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[142] Jochen Wengenroth. *Wahrscheinlichkeitstheorie.* Walter de Gruyter, Berlin, 2008.

[143] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012.

[144] Burton Wu, Clare A. Mcgrory, and Anthony N. Pettitt. A New Variational Bayesian Algorithm with Application to Human Mobility Pattern Modeling. *Statistics and Computing*, 22(1):185–203, January 2012.

[145] C. F. J. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, 1983.

[146] Chong Wu, Can Yang, Hongyu Zhao, and Ji Zhu. On the convergence of the EM algorithm: A data-adaptive analysis. *arXiv preprint, arXiv:1611.00519*, 2016.

[147] Lei Xu and Michael I. Jordan. On Convergence Properties of the Em Algorithm for Gaussian Mixtures. *Neural Comput.*, 8(1):129–151, jan 1996.

[148] Hongyi Zhang, Sashank J. Reddi, and Suvrit Sra. Riemannian SVRG: Fast Stochastic Optimization on Riemannian Manifolds. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[149] D. Zoran and Y. Weiss. Natural images, Gaussian mixtures and dead leaves. *Advances in Neural Information Processing Systems*, 3:1736–1744, 01 2012.

APPENDIX A

Foundations of Differential Geometry and Statistics

A.1 Foundations of Differential Geometry

In this section, we state relevant definitions and differential geometry that are mentioned in Chapter 2. The definitions and theorems presented here can be found in [2].

Definition A.1. (topology) *A* topology *on a set M is a collection of $\mathcal{T}$ subsets of M, called* open sets*, such that*

1. *M and $\emptyset$ are in $\mathcal{T}$,*
2. *the union of elements of any subcollection of $\mathcal{T}$ is in $\mathcal{T}$ and*
3. *the intersection of the elements of any finite subcollection of $\mathcal{T}$ is in $\mathcal{T}$.*

A subset C of M is called closed *if it is the complement of an open set in M, that is, $M \setminus C$ is open. In particular, M and $\emptyset$ are both open and closed. A* topological space *is a tuple $(M, \mathcal{T})$, where M is a set and $\mathcal{T}$ is a topology on M.*

Definition A.2. (atlas topology) *Given a maximal atlas $\mathcal{A}^+$ on a set M, the* atlas topology *on M states that a subset of M is open if and only if it is the union of collection of chart domains.*

Definition A.3. (basis for a topology) *A collection $\mathcal{B}$ of subsets of M is a* basis *for a topology on M if*

1. *for each $x \in M$, there exists $B \in \mathcal{B}$ such that $x \in B$,*

2. *if $x \in B_1 \cap B_2 for B_1, B_2 \in \mathcal{B}$, there exists $B_3 \in \mathcal{B}$ and $B_3 \subset B_1 \cap B_2$.*

If $\mathcal{B}$ is a basis for a topology $\mathcal{T}$ on M, then $\mathcal{T}$ equals the collection of all unions of elements of $\mathcal{B}$.

In the definition of a manifold (Definition 2.4), the following assumptions are made regarding the atlas topology.

Definition A.4. (second-countable) *A topology $\mathcal{T}$ on a set M is called* second-countable *if there is a countable basis for $\mathcal{T}$.*

Definition A.5. (Hausdorff) *A topology $\mathcal{T}$ on a set M is called* Hausdorff *if all pairs of distinct points have disjoint neighborhoods, that is: for all $\theta \neq \theta^{'}$ in M there exist open sets O and $O^{'}$ such that $\theta \in O, \theta \in O^{'}$ and $O \cap O^{'} = \emptyset$.*

The following results give us the desired properties for performing optimization on possibly non-euclidean spaces. Studying convergence to local minima is meaningful with the Hausdorff property and the existence of a Riemannian metric is given by second-countability of the atlas topology.

Theorem A.6. *[120, Prop. 2.5] Let $\mathcal{T}$ be a topology on M. If $\mathcal{T}$ is Hausdorff, then every sequence of points of M converges to at most one limit point of M.*

Theorem A.7. *[81, Prop. 13.3] Every smooth manifold $\mathcal{M}$ admits a Riemannian metric.*

In this thesis, we consider product manifolds. For reasons of completeness, we state the definition of a product topology.

Definition A.8. (product topology) *Let $(M_1, \mathcal{T}_1)$, $(M_2, \mathcal{T}_2)$ be topological spaces. A family of subsets of $M_1 \times M_2$ defined by $\mathcal{B} := \{U_1 \times U_2 | U_1 \in \mathcal{T}_1, U_2 \in \mathcal{T}_2\}$ is an open basis of a topology $\mathcal{T}$ of $M_1 \times M_2$. The topology $\mathcal{T}$ and topological space $(M_1 \times M_2, \mathcal{T})$ are called the* product topology *and* product space*, respectively.*¸

A.2 Brief introduction to Bayesian Statistics

In Chapter 5, we introduce a penalization term according to [66] that is based on Bayesian statistics. We briefly introduce the basic idea of Bayesian statistics required for a deeper understanding for the motivation of maximum a posterior estimation. We consider the statistical setting as in Section 3.1. In this context, we consider the (factorized) *conditional expectation.*

Definition A.9. (conditional expectation) *[142, Chapter 6] Let X, Y be random variables and assume their joint probability density function is given by $f_{X,Y}$. Further, we denote the marginal distribution of X by f_X. Then, the* conditional probability density function at $X = x$ *of Y, $f_{Y|X=x}$ is defined by*

$$f_{Y|X=x}(y|x) = \frac{f_{X,Y}(x,y)}{f_X(x)},$$

if $f_X(x) > 0$. In case $f_X(x) = 0$, we set $f_{Y|X=x}(y|x) = 0$ for all y.
The factorized conditional expectation $\mathbb{E}(Y|X = x)$ *is given by*

$$\mathbb{E}(Y|X = x) = \int_{-\infty}^{\infty} y f_{Y|X=x}(y|x)dy.$$

A parametric estimation approach can be further divided into a *frequentist* and a *Bayesian* approach. In the frequentist approach, we assume that the parameter of interest θ is a deterministic parameter and we estimate its value (point estimation). Contrary, in a Bayesian setting, we assume that the parameter θ comes itself with uncertainty, that is it is a random variable itself. The basis of Bayesian statistics is the well-known Bayes formula:

Theorem A.10. *[86] Let X, Y be random variables with probability density functions f_X, f_Y, respectively. For the conditional probability density function of $f_{X|Y=y}$, it holds that*

$$f_{X|Y=y}(x) = \frac{f_{Y|X=x}(y) f_X(x)}{f_Y(y)},$$

where $f_{Y|X=x}(y)$ is the conditional probability density function of $Y|X = x$.

In Bayesian statistics, we equip the parameter of interest θ with a prior belief, that is we impose a *prior* probability density function $g_\theta(\theta)$ reflecting our initial belief about the value of θ. After an experiment, we have observed data $x := x_1, \ldots, x_n$ and we assume that this data has been generated by a probability density function parameterized by θ, that is $X_i \sim f_X(x^v|\theta)$ and x_i is a realization of the random variable X_i. With these samples, we can update our belief about θ yielding the *posterior* probability density function. According to Bayes' theorem, it holds that

$$g_{\theta|X=x}(\theta) = c f_X(x|\theta) g_\theta(\theta),$$

where c is a normalizing constant. Here, $g_\theta(\theta)$ is called the *prior probability density function of θ* and $g_{\theta|X=x}(\theta)$ is called the *posterior probability density function of θ given*

x. The choice of the prior $g_\theta(\theta)$ has a big impact on the posterior $g_{\theta|X=x}$. A special choice of the prior is a conjugate prior: We say g_θ is a *conjugate prior* for f_X if the prior g_θ and the posterior $g_{\theta|X=x}$ are in the same distribution family. The choice of a conjugate prior is very convenient for applications as it comes with a closed-form expression for the posterior.

APPENDIX B

Appendix for Gaussian Mixture Models

B.1 Classical EM for GMMs

At this place, we derive the Expectation Maximization algorithm for Gaussian mixture models due to its centrality to Chapter 5. We start with the classical algorithm and then with a penalized version that is in line with the penalized objective (5.13). The derivation presented here can be found in [101, Section 11.4.2] and [94, Section 2.7].

We here show how the maximum likelihood problem (5.3) can be described with latent variables as in Section 3.2.

Assume we have a clustering tasks with K clusters denoted by C_j, $j = 1, \dots, K$, and data in each cluster follows a Gaussian distribution with mean μ_j and covariance matrix Σ_j.We introduce hidden iid variables $Z_1, \dots, Z_n$ modeling the class membership, that is the variable Z_i takes on the value j if the corresponding X_i is in cluster C_j, $j = 1, \dots, K$ with probability α_j. Thus, we assume that Z_i follows a (discrete) categorical distribution, that is $Z_i \sim Cat(\alpha)$. Hence, the probability mass function of Z_i reads

$$f_{Z_i}(z_i^v = j) = \alpha_j \quad \text{for } j = 1, \dots, K$$

or equivalently

$$f_{Z_i}(z_i^v) = \prod_{j=1}^{K} (\alpha_j)^{\mathbb{1}_{\{z_i^v = j\}}}$$

The likelihood function (5.2) of observations $x_1, \dots, x_n$ can thus be expressed as

$$\begin{aligned} l(\tilde{\theta}) &= \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} \alpha_j p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j) \right) \\ &= \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} f_{Z_i}(z_i^v = j) p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j) \right) \\ &= \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} f_{X_i, Z_i = j}(x_i, z_i; \theta) \right). \end{aligned}$$

The setting described in Section 3.2 is thus met. We now concretize the E-step and the M-step of the EM algorithm (Algorithm 7).
For $\tilde{\theta} = (\tilde{\theta}_1, \dots, \tilde{\theta}_K)$, where $\tilde{\theta}_j = (\alpha_j, \mu_j, \Sigma_j)$, we observe that

$$\begin{aligned} Q(\tilde{\theta}, \tilde{\theta}^t) &= \sum_{i=1}^{n} \mathbb{E}_{Z_i | X_i = x_i; \tilde{\theta}^t} \left[\log \left(\prod_{j=1}^{K} \alpha_j p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j) \right)^{\mathbb{1}_{\{Z_i = j\}}} \right] \\ &= \sum_{i=1}^{n} \sum_{j=1}^{K} \mathbb{E}_{Z_i | X_i = x_i; \tilde{\theta}^t} \left[\mathbb{1}_{\{Z_i = j\}} \right] \log \left(\alpha_j p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j) \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{K} f_{Z_i | X_i = x_i}(z_i^v | x_i; \tilde{\theta}^t) \left(\log(\alpha_j) + \log \left(p_{\mathcal{N}}(x_i; \mu_j, \Sigma_j) \right) \right). \end{aligned} \tag{B.1}$$

We denote the *responsibility of cluster j for observation x_i* at iteration t by r_{ij}^t and get by using Bayes' formula

$$r_{ij}^t := f_{Z_i | X_i = x_i}(z_i^v | x_i; \tilde{\theta}^t) = \frac{\alpha_j p_{\mathcal{N}}(x_i; \mu_j^t, \Sigma_j^t)}{\sum\limits_{k=1}^{K} \alpha_k p_{\mathcal{N}}(x_i; \mu_k^t, \Sigma_k^t)}. \tag{B.2}$$

The computation of the E-step thus consists in evaluating the responsibility (B.2) for each observation x_i. For the M-step, we maximize the function (B.1) with respect to $\tilde{\theta}$, that is with respect to α_j, μ_j and Σ_j for all $j = 1, \dots, K$. Luckily, this is a concave problem with a closed form solution. We state the algorithm in the following.

Algorithm 8: Expectation Maximization algorithm for Gaussian mixture models

```
  Input: initial values for $\tilde{\theta}^0$
  Output: sequence of parameters $\{\alpha_j^t, \mu_j^t, \Sigma_j^t\}$ for $j = 1, \dots, K$
1 for $t = 0, 1, 2, \dots$ do
2     E-step: Evaluate
          $$r_{ij}^t = \frac{\alpha_j p_{\mathcal{N}}(x_i; \mu_j^t, \Sigma_j^t)}{\sum_{k=1}^{K} \alpha_k p_{\mathcal{N}}(x_i; \mu_k^t, \Sigma_k^t)};$$
      for $i = 1, \dots, n$ and $j = 1, \dots, K$.
3     M-step: For $j = 1, \dots, K$, set
          $$\alpha_j^t = \frac{1}{n} \sum_{i=1}^{n} r_{ij}^t;$$
          $$\mu_j^t = \frac{1}{\alpha_j^t} \sum_{i=1}^{n} r_{ij}^t x_i;$$
          $$\Sigma_j^t = \frac{1}{\alpha_j^t} \sum_{i=1}^{n} r_{ij}^t (x_i - \mu_j)(x_i - \mu_j)^T;$$
4 end for
```

B.2 Penalized EM for GMMs

We here briefly state the derivation of the penalized EM algorithm for Gaussian mixture models as presented in [106]. As mentioned in Section 5.2.1, a penalization term can be found by considering the a posterior log-likelihood which is the sum of the classical (frequentist) log-likelihood $\mathcal{L}(\theta)$ and the logarithm of the prior probability density function for θ. In this section, we specify the prior probability density functions arising in the maximum a posterior likelihood (5.13).

Since the single components of θ, i.e. α, μ_j, Σ_j, $j = 1, \dots, K$ are independent from each other, the prior g_θ of θ is the product of the individual priors denoted by g_α, g_{μ_j} and g_{Σ_j}. As priors, conjugate priors are a favorable choice for g_α, g_μ, g_Σ as they allow for beneficial EM update rules. As introduced in Section 3.1, we denote arguments of the densities g_α, g_{μ_j} and g_{Σ_j} by α^v, μ_j^v and Σ_j^v, respectively.

A conjugate prior for $\alpha = (\alpha_1, \ldots, \alpha_K)$ is the (symmetric) Dirichlet distribution whose probability density function is given by

$$g_{\alpha}^{dirich}(\alpha^v; \zeta) = C_1(\zeta) \prod_{j=1}^{K} (\alpha_j^v)^{\zeta},$$

where $\zeta > 0$ is known and $C_1(\zeta) > 0$ is a normalizing constant independent of α [106]. For each parameter μ_j , the conjugate prior is given by a Gaussian distribution $p_{\mathcal{N}}$ with mean ν. Its covariance matrix is given by $\kappa^{-1}\Sigma_j$, where $\kappa > 0$ is a shrinkage parameter. The conjugate prior for the means μ_j this reads

$$g_{\mu_j}^{\mathcal{N}}(\mu_j^v; \Sigma_j, \kappa, \lambda) = C_2(\kappa) \det(\Sigma_j)^{-1/2} \exp\left(-\frac{\kappa}{2}(\mu_j^v - \lambda)\Sigma_j^{-1}(\mu_j^v - \lambda)^T\right),$$

where $\kappa > 0$, $\lambda \in \mathbb{R}^d$ and $C_2(\kappa) > 0$ is a constant independent of μ_j^v, Σ_j, where Σ_j is assumed to be given [106]. The conjugate prior for the covariance matrices Σ_j is an inverse Wishart distribution, that is

$$g_{\Sigma_j}^{Wi}(\Sigma_j^v; \Lambda, \nu) = C_3(\Lambda, \nu) \det(\Sigma_j^v)^{-(\nu+d+1)/2} \exp\left(-\frac{1}{2} \operatorname{tr}\left(\Lambda(\Sigma_j^v)^{-1}\right)\right),$$

where $C_3(\Lambda, \nu) > 0$ is a parameter independent of $(\Sigma_j^v)^{-1}$. Here, $\Lambda \in \mathbb{R}^{d \times d}$ is a symmetric positive definite matrix and $\nu > 0$. [1]

Thus, the a posterior log-likelihood function for Gaussian mixture models is given by the expression (5.13). For this objective, one can formulate a penalized version of the EM algorithm (Algorithm 8). For a derivation of the algorithm, we refer to [106, Appendix 1]. Thanks to the conjugate priors, the E-step remains the same as in Algorithm 9, only the M-step changes. We briefly state the algorithm in the following.

[1]We impose the same parameters for the conjugate priors of μ_j, Σ_j, $j = 1, \ldots, K$, meaning that the parameters $\kappa, \lambda, \Lambda, \nu$ are independent of j.

Algorithm 9: Penalized Expectation Maximization algorithm for Gaussian mixture models

```
  Input: initial values for $\tilde{\theta}^0$
  Output: sequence of parameters $\{\alpha_j^t, \mu_j^t, \Sigma_j^t\}$ for $j = 1, \dots, K$
1 for $t = 0, 1, 2, \dots$ do
2     E-step: Evaluate
          $$r_{ij}^t = \frac{\alpha_j p_{\mathcal{N}}(x_i; \mu_j^t, \Sigma_j^t)}{\sum_{k=1}^{K} \alpha_k p_{\mathcal{N}}(x_i; \mu_k^t, \Sigma_k^t)};$$
      for $i = 1, \dots, n$ and $j = 1, \dots, K$.
3     M-step: For $j = 1, \dots, K$, set
          $$\alpha_j^t = \frac{\sum_{i=1}^{n} r_{ij}^t + \zeta}{n + K\zeta};$$
          $$\mu_j^t = \frac{\sum_{i=1}^{n} r_{ij}^t x_i + \kappa\lambda}{\sum_{i=1}^{n} r_{ij}^t + \kappa};$$
          $$\Sigma_j^t = \frac{\sum_{i=1}^{n} r_{ij}^t (x_i - \mu_j)(x_i - \mu_j)^T + \kappa(\mu_j - \lambda)(\mu_j - \lambda)^T + \Lambda}{\sum_{i=1}^{n} r_{ij}^t + \nu + d + 2};$$
4 end for
```

B.3 Convergence of R-NTR to Critical Points for Gaussian Mixture Models

In Section 5.3, we introduced assumptions needed to show convergence of Algorithm 3 to a critical point for Gaussian mixture models, see Theorem 5.8. We prove the Theorem in the following.

Proof of Theorem 5.8.
According to Theorem 2.31, it suffices to show that the sequence of iterates θ^t remains

bounded. Since the rejection threshold in Algorithm 3 is nonnegative, i.e. $\rho' > 0$, we get

$$\hat{\mathcal{L}}_{\text{pen}}(\theta^0) \leq \hat{\mathcal{L}}_{\text{pen}}(\theta^t) \tag{B.3}$$

for all iterations $t = 0, 1, 2, \ldots$.

We show that the iterates S_j^t remain in the interior of $\mathbb{P}^{d+1}$. For this, assume there exists a subsequence θ^{t_i} with $\lambda_{\min}(S_j^{t_i}) \to 0$ as $t_i \to \infty$. By the proof of Theorem 5.5, this implies $\hat{\mathcal{L}}_{\text{pen}}(\theta^{t_i}) \xrightarrow{t_i \to \infty} -\infty$ which is a contradiction to (B.3). Thus, there exists a lower bound $C_l > 0$ such that

$$C_l \leq \lambda_{\min}(S_j^t). \tag{B.4}$$

For the upper bound, we consider the set of successful (unsuccessful) steps $\mathfrak{S}_t$ ($\mathfrak{F}_t$) generated by the algorithm until iteration t given by

$$\mathfrak{S}_t = \{l \in \{0, 1, \ldots, t\} : \rho_l > \rho'\}, \qquad \mathfrak{F}_t = \{l \in \{0, 1, \ldots, t\} : \rho_l \leq \rho'\}.$$

Let

$$\xi^t = \left(\left(\xi_{S_1}^t, \ldots, \xi_{S_K}^t\right), {\xi_\eta}^t\right) \in T_\theta \mathcal{M}_{GMM}$$

be the tangent vector returned by solving the quadratic subproblem in line 2, Algorithm 3 and $R_{S_j^t}(\xi_{S_j^t}) = S_j^t \exp\left((S_j^t)^{-1}\xi_{S_j^t}\right)$ be the retraction of ξ^t at iteration t with respect to S_j^t, see (5.27). Due to the boundedness of the quadratic subproblem in (3), there exists $\tilde{\Delta} > 0$ such that $\|{\xi_{S_j}}^t\| \leq \tilde{\Delta}$ for all $j = 1, \ldots, K$. We get

$$\begin{aligned}
\|\xi_{S_j}^t\| &= \|R_{S_j^t}(\xi_{S_j^t})\|\mathbb{1}_{\{t\in\mathfrak{S}_t\}} + \|S_j^{t-1}\|\mathbb{1}_{\{t\in\mathfrak{F}_t\}} \\
&\leq \|S_j^{t-1}\| \left(\exp\left(\|\left(S_j^{t-1}\right)^{-1}\|\|\xi_{S_j}^t\|\right)\mathbb{1}_{\{t\in\mathfrak{S}_t\}} + \mathbb{1}_{\{t\in\mathfrak{F}_t\}}\right) \\
&\leq \|S_j^{t-1}\| \left(\mathbb{1}_{\{t\in\mathfrak{F}_t\}} + \exp\left(\frac{\|\xi_{S_j^t}\|}{\lambda_{\min}(S_j^{t-1})}\right)\mathbb{1}_{\{t\in\mathfrak{S}_t\}}\right) \\
&\leq \|S_j^{t-1}\| \left(\mathbb{1}_{\{t\in\mathfrak{F}_t\}} + \exp\left(\frac{\tilde{\Delta}}{C_l}\right)\mathbb{1}_{\{t\in\mathfrak{S}_t\}}\right) \\
&\leq \tau^{t-1}\|\Psi\| \left(\mathbb{1}_{\{t\in\mathfrak{F}_t\}} + \exp\left(\frac{\tilde{\Delta}}{C_l}\right)\mathbb{1}_{\{t\in\mathfrak{S}_t\}}\right)
\end{aligned}$$

and from the assumption (5.41) boundedness from above follows directly.

We now show boundedness of

$$\eta^t = (\eta_1^t, \ldots, \eta_{K-1}^t) \in \mathbb{R}^{K-1}.$$

By the inequality (B.3) and (5.17), we have

$$\hat{\mathcal{L}}_{\text{pen}}(\theta^0) \leq \hat{\mathcal{L}}_{\text{pen}}(\theta^t) = \hat{\mathcal{L}}(\theta^t) + \sum_{j=1}^{K} \psi(S_j^t, \Psi) + \varphi(\eta^t, \zeta).$$

Due to (B.4), there exists $\tilde{C} > 0$ such that

$$\hat{\mathcal{L}}_{\text{pen}}(\theta^0) \leq \hat{\mathcal{L}}_{\text{pen}}(\theta^t) \leq \tilde{C} + \varphi(\eta^t, \zeta). \tag{B.5}$$

We will study $\varphi(\eta^t, \zeta)$ for η^t at the boundary in the following. We distinguish between the following two cases:

I. Assume there exists $j \in \{1, \dots, l\}$ such that $\eta_j^t \to \infty$.

II. Assume there does not exist a $j \in \{1, \dots, K-1\}$ such that $\eta_j^t \to \infty$.

We first study the first case, that is I. For this, we distinguish two cases:

I. a) We study the case where only some of the η_j^t approach ∞. Without loss of generality, assume that $\eta_j^t \to \infty$ for $j \in \{1, \dots, l\}$ for $l < K-1$. Further, for all $j > l$, assume that $\eta_j^t \leq c$ for a constant $c > 0$.
Recall that

$$\alpha_j^t = \frac{\exp(\eta_j^t)}{\sum\limits_{k=1}^{K} \exp(\eta_k^t)}$$

for all $j = 1, \dots, K$ with $\eta_K = 0$ by (5.8). By assumption (5.40), we have $\alpha_j^t > \epsilon$. By the rule of L'Hôpital, we get

$$\lim_{t\to\infty} \alpha_j^t = \lim_{t\to\infty} \frac{\exp(\eta_j^t)}{\sum\limits_{j=1}^{K} \exp(\eta_k^t)} = \frac{1}{l},$$

yielding

$$\lim_{t\to\infty} \sum_{j=1}^{K} \alpha_j^t = 1 + \sum_{j=l+1}^{K} \lim_{t\to\infty} \alpha_j^t \geq 1 + \sum_{j=l+1}^{K} \epsilon_j > 1$$

which is a contradiction to $\sum\limits_{j=1}^{K} \alpha_j = 1$, see (5.4).

I. b) Now assume that for all $j = 1, \dots, K-1$ we have $\eta_j^t \to \infty$ with the same speed, otherwise we are in case I.a). Without loss of generality, we set $n = \eta_j^t$ and let $n \to \infty$. Using $\eta_K = 0$ and $\exp(n) > -1$, the penalization term $\varphi(\eta^t, \zeta)$ from (5.16) reads

$$\begin{aligned}
\varphi(\eta^t, \zeta) &= \zeta \left(\sum_{j=1}^{K} \eta_j^t - K \log \left(\sum_{k=1}^{K} \exp(\eta_k^t) \right) \right) \\
&= \zeta \left((K-1)n - K \log \left(1 + \sum_{k=1}^{K-1} \exp(n) \right) \right) \\
&\leq \zeta \left((K-1)n - K \log(K-2) - K \exp(n) \right) \\
&= \zeta(-n - K \log(K-2)) \xrightarrow[n \to \infty]{} -\infty
\end{aligned}$$

which is a contradiction to (B.5).

We now study case II., that is we assume $\nexists j \in \{1, \dots, K-1\} : \eta_j^t \to \infty$. For this, assume $\exists l \leq K-1 : j \in \{1, \dots, l\} : \eta_j^t \to -\infty$ and for $j > l : |\eta_j^t| \leq c$. Then, the penalization term (5.16) reads

$$\begin{aligned}
\varphi(\eta^t, \zeta) &= \zeta \left(\sum_{j=1}^{l} \eta_j^t + \sum_{j=l+1}^{K-1} \eta_j^t - K \log \left(\sum_{k=1}^{K} \exp(\eta_k^t) \right) \right) \\
&\leq \zeta \left(\sum_{j=1}^{l} \eta_j^t + (K-1-l)c \right) \xrightarrow[\eta_j^t \to -\infty]{} -\infty,
\end{aligned}$$

where we used $\eta_K = 0$. As before, this is a contradiction to (B.5).

Thus, the iterates $\{\theta\}^t$ remain in a compact set and the assumptions of Theorem 2.32 are fulfilled.

□

B.4 Additional results for Gaussian Mixture Models

We show additional numerical results for the experiments conducted in Section 5.4.1 for both simulated and additional real-world data sets.

B.4.1 Simulated Data

Table B.1: Simulation results of 20 runs for dimensions $d = 20$, number of components $K = 5$ and eccentricity $e = 5$.

		EM	R-NTR	R-LBFGS
c=0.2	Iterations	145.7	38.5	60.4
	Mean time (s)	1.899	1.512	8.165
	Mean ALL	-50.65	-50.66	-50.66
	MSE weights	0.00018	0.00019	0.008
	MSE means	0.08	0.08	0.08
	MSE cov	2.18	1.83	2.21
c=1	Iterations	258.9	60.4	84.1
	Mean time (s)	3.061	1.951	11.64
	Mean ALL	-57.65	-57.65	-57.65
	MSE weights	0.00037	0.0004	0.008
	MSE means	0.08	0.07	0.07
	MSE cov	1.02	1.39	1.04
c=5	Iterations	194.6	37.2	65.7
	Mean time (s)	2.367	1.267	8.927
	Mean ALL	-54.86	-54.86	-54.87
	MSE weights	0.00017	0.00018	0.008
	MSE means	0.07	0.07	0.07
	MSE cov	0.15	0.13	0.15

Table B.2: Simulation results of 20 runs for dimensions $d = 40$, number of components $K = 5$ and eccentricity $e = 10$ for different number of observations

(a) $n = 1000$ observations

		EM	R-NTR	R-LBFGS
c=0.2	Iterations	4	16.3	6.6
	Mean time (s)	0.091	1.336	0.622
	Mean ALL	-125.63	-125.63	-125.649
	MSE weights	3e-05	3e-05	0.008
	MSE means	0.091	0.095	0.091
	MSE cov	0.224	0.215	0.224
c=1	Iterations	4	17.6	8.8
	Mean time (s)	0.097	1.561	0.862
	Mean ALL	-110.112	-110.112	-110.14
	MSE weights	3e-05	3e-05	0.008
	MSE means	0.089	0.084	0.089
	MSE cov	0.461	0.337	0.461
c=5	Iterations	4	17	7.2
	Mean time (s)	0.094	1.497	0.697
	Mean ALL	-116.276	-116.276	-116.3
	MSE weights	3e-05	3e-05	0.008
	MSE means	0.061	0.066	0.061
	MSE cov	0.351	0.43	0.351

(b) $n = 10.000$ observations

		EM	R-NTR	R-LBFGS
c=0.2	Iterations	4	2.3	3.3
	Mean time (s)	0.628	1.524	0.74
	Mean ALL	-127.762	-127.762	-127.764
	MSE weights	0	0	0.008
	MSE means	0.089	0.09	0.089
	MSE cov	0.14	0.189	0.14
c=1	Iterations	4	2.8	3.8
	Mean time (s)	0.609	1.676	0.884
	Mean ALL	-112.25	-112.25	-112.252
	MSE weights	0	0	0.008
	MSE means	0.084	0.084	0.084
	MSE cov	0.313	0.29	0.313
c=5	Iterations	4	2.6	3.6
	Mean time (s)	0.63	1.662	0.774
	Mean ALL	-118.337	-118.337	-118.339
	MSE weights	0	0	0.008
	MSE means	0.065	0.071	0.065
	MSE cov	0.228	0.235	0.228

B.4.2 Real-world Data

Gas Turbine CO and NOx Emission Data Set [76]

Table B.3: Results of (normalized) Gas turbine CO and NOx Emission Data Set for different number of components K. Number of observations $n = 36733$, dimensions $d = 11$.

		EM	R-NTR	R-LBFGS
K = 2	Time (s)	2.04	2.48	3.22
	Iterations	25	22	28
	ALL	-5.73	-5.73	-5.73
K = 5	Time (s)	17.41	9.52	11.48
	Iterations	90	46	58
	ALL	-1.93	-1.99	-1.93
K = 10	Time (s)	40.66	29.36	27.31
	Iterations	130	61	75
	ALL	-1.17	-1.16	-1.17
K = 15	Time (s)	52.06	63.20	142.77
	Iterations	115	76	233
	ALL	0.04	-0.04	-0.05

Wine Quality Data set [35]

Table B.4: Results of (normalized) Wine quality data set for different number of components K. Number of observations $n = 6497$, dimensions $d = 11$.

		EM	R-NTR	R-LBFGS
K = 2	Time (s)	0.54	0.24	1.55
	Iterations	27	8	20
	ALL	-11.02	-11.02	-11.02
K = 5	Time (s)	4.75	1.24	5.69
	Iterations	154	34	51
	ALL	-9.74	-9.98	-9.88
K = 10	Time (s)	12.61	5.65	22.80
	Iterations	239	83	100
	ALL	-9.23	-9.28	-9.28
K = 15	Time (s)	91.16	11.17	51.65
	Iterations	1137	70	147
	ALL	-8.91	-8.88	-8.89

The wine quality data set also provides classification labels: we can distinguish between white and red wine or distinguish between 7 quality labels. Besides the clustering performance of the methods (Table B.5), we also show the goodness of fit of our method for $K = 2$ and $K = 7$.

Table B.5: Weighted mean squared errors of (normalized) Wine data set for $K = 2$ (white wine/ red wine).

	EM	R-NTR	R-LBFGS
distance	2.757966	2.757982	2.757974
wMSE weight	0.003169	0.003169	0.003169
wMSE mean	0.073776	0.073779	0.073778
wMSE cov	0.562106	0.562113	0.562109

Table B.6: Adjusted Rand Index for (normalized) Wine data set for $K = 2$ (white wine/ red wine).

	ground truth	EM	R-NTR	R-LBFGS
ground truth	1	0.78	0.78	0.78
EM		1	1.00	1.00
R-NTR			1	1.00
R-LBFGS				1

Table B.7: Weighted mean squared errors of (normalized) Wine data set for $K = 7$ (quality label). Weighting by mixing coefficient of the mixing coefficients belonging to the true labels.

	EM	R-NTR	R-LBFGS
distance	89.259333	89.488919	88.894978
wMSE weight	0.019019	0.010065	0.035496
wMSE mean	3.442393	4.214438	3.082073
wMSE cov	12.047944	13.400811	11.327231

Table B.8: Weighted mean squared errors of (normalized) Wine data set for $K = 7$ (quality label). Weighting by mixing coefficients of the respective method.

	EM	R-NTR	R-LBFGS
distance	89.259333	89.488919	88.894978
wMSE weight	0.019019	0.010065	0.035496
wMSE mean	6.068149	5.617490	11.996024
wMSE cov	61.808676	37.060684	55.707085

Table B.9: Adjusted Rand Index for (normalized) Wine data set for $K = 7$ (quality label).

	ground truth	EM	R-NTR	R-LBFGS
ground truth	1	0.02	0.01	0.02
EM		1	0.73	0.70
R-NTR			1	0.85
R-LBFGS				1