Mathematical Programming (2020) 181:187–223
https://doi.org/10.1007/s10107-019-01381-4

FULL LENGTH PAPER

Series A

An alternative to EM for Gaussian mixture models: batch and stochastic Riemannian optimization

Reshad Hosseini[1,2] · Suvrit Sra[3]

Received: 9 June 2017 / Accepted: 17 February 2019 / Published online: 19 March 2019

Abstract

We consider maximum likelihood estimation for Gaussian Mixture Models (GMMs). This task is almost invariably solved (in theory and practice) via the Expectation Maximization (EM) algorithm. EM owes its success to various factors, of which is its ability to fulfill positive definiteness constraints in closed form is of key importance. We propose an alternative to EM grounded in the Riemannian geometry of positive definite matrices, using which we cast GMM parameter estimation as a Riemannian optimization problem. Surprisingly, such an out-of-the-box Riemannian formulation completely fails and proves much inferior to EM. This motivates us to take a closer look at the problem geometry, and derive a better formulation that is much more amenable to Riemannian optimization. We then develop Riemannian batch and stochastic gradient algorithms that outperform EM, often substantially. We provide a non-asymptotic convergence analysis for our stochastic method, which is also the first (to our knowledge) such global analysis for Riemannian stochastic gradient. Numerous empirical results are included to demonstrate the effectiveness of our methods.

Keywords Stochastic optimization · Riemannian optimization · Gaussian mixture models · Positive definite matrices · Retraction · Non-asymptotic rate of convergence

Mathematics Subject Classification 49 · 53 · 58 · 62 · 90

A preliminary version of this work appeared at the *Advances in Neural Information Processing Systems (NIPS 2015)*, wherein this reformulation was originally introduced.

S. Sra was partially supported by NSF-IIS-1409802.

✉ Reshad Hosseini
reshad.hosseini@ut.ac.ir

Suvrit Sra
suvrit@mit.edu

1 School of ECE, College of Engineering, University of Tehran, Tehran, Iran

2 School of Computer Science, Institute of Research in Fundamental Sciences (IPM), Tehran, Iran

3 Massachusetts Institute of Technology, Cambridge, MA, USA

1 Introduction

Gaussian Mixture Models are extensively used across many tasks in machine learning, signal processing, and other areas [7,14,15,24,27,29,33]. For a vector $\boldsymbol{x} \in \mathbb{R}^d$, the density of a Gaussian Mixture Model (GMM) is given by

$$p(\boldsymbol{x}) := \sum_{j=1}^{K} \alpha_j p_{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j), \tag{1}$$

where $p_{\mathcal{N}}$ is a Gaussian density with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma} \succ 0$, i.e.,

$$p_{\mathcal{N}}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) := \det(\boldsymbol{\Sigma})^{-1/2}(2\pi)^{-d/2} \exp\big(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\big).$$

Given i.i.d. samples $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ drawn from (1), we seek maximum likelihood estimates $\{\hat{\boldsymbol{\mu}}_j \in \mathbb{R}^d, \hat{\boldsymbol{\Sigma}}_j \succ 0\}_{j=1}^K$ and $\hat{\boldsymbol{\alpha}} \in \Delta_K$ of the parameters of the GMM. This estimation is cast as the following log-likelihood maximization problem:

$$\max_{\boldsymbol{\alpha} \in \Delta_K, \{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \succ 0\}_{j=1}^K} \quad \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} \alpha_j p_{\mathcal{N}}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) \right). \tag{2}$$

A quick literature search reveals that (2) is most frequently solved via the Expectation Maximization (EM) algorithm [13] or its variants. Although other optimization methods have also been considered [32], for solving practical instances of (2) usual methods such as conjugate gradients, quasi-Newton, Newton, are often regarded as being inferior to EM [44].

Difficulties and motivation The primary reason why standard nonlinear methods have difficulties in solving (2) is the positive definiteness constraint on the covariance matrices. Since this constraint defines an open subset of Euclidean space, in principle, if the iterates remain in the interior, standard unconstrained Euclidean optimization methods could be used. The iterates may, however, approach the boundary of the constraint set, especially in higher dimensions, which can lead to very slow convergence. For instance, a possible approach is to formulate the positive definite constraint via a set of smooth convex inequalities [40] and use interior-point methods. It was observed in [37] that using such sophisticated methods can be vastly slower (on some closely related statistical problems) than simpler EM-like fixed-point methods, especially with growing problem dimensionality.

Another "natural" approach to handle the positive definite constraint is to use the Cholesky decomposition, as was exploited for semidefinite programming in [11], and more recently in [6]. In general, this decomposition can add spurious local maxima and stationary points to the objective function of general optimization problems, even for convex semidefinite programs [40]. Remarkably, it can be shown that such a decomposition does not add spurious local maxima to (2). Nevertheless, we observed (empirically) that the convergence speed of standard nonlinear solvers for estimating parameters of (2) using Cholesky decomposition is considerably slower than EM.

Motivated by the success of non-Euclidean optimization for some problems with positive definite variables [37,38], we consider an alternative approach to EM. In particular, we solve (2) via *Riemannian optimization*. Surprisingly, a naïve use of Riemannian methods completely fails to compete with EM, while their use on a careful reformulation of (2) demonstrably succeeds.

We describe this reformulation in Sect. 3, and remark here informally on why a naïve use of manifold optimization fails: The negative log-likelihood for a single Gaussian is Euclidean convex (the key property that makes the "M-step" of EM easy), but not geodesically convex. Reformulating the problem to remove this geometric mismatch might therefore be fruitful, i.e., if we reformulate the single Gaussian likelihood to be geodesically convex, manifold optimization may benefit. This intuition turns out to have remarkable empirical consequences as will become apparent from the paper.

This reformulation was originally introduced in the preliminary work [20]. However, this preliminary work lacked global convergence results and did not present any stochastic optimization algorithm. Moreover, in this preliminary work it was impossible to develop a global convergence result starting from an arbitrary point, because the log-likelihood of GMM is not bounded above. To resolve this difficulty, in this paper we develop similar reformulation for a penalizer that helps the objective function be bounded above. We further present a global convergence analysis for the Riemannian stochastic gradient descent (SGD) for optimizing the penalized objective function.

Contributions The main contributions of this paper are the following:

- In our preliminary work [20], we developed a reformulation only for GMMs. In the current paper, we develop a reformulation for richer likelihood models that incorporate conjugate priors.
- In our preliminary work [20], we presented batch optimization algorithms. In this paper, we present a stochastic optimization algorithm that greatly enhances the scalability of our methods. It is worth mentioning, our methods permit the use of retractions (beyond the usual exponential map) and vector transport, which enables further scalability.
- We provide an iteration complexity analysis of stochastic gradient on manifolds, obtaining a $O(1/\sqrt{T})$ bound. To our knowledge, this is the first *non-asymptotic* convergence analysis for stochastic gradient on manifolds. Subsequently, we present analysis that outlines why Riemannian SGD applies to penalized GMM-likelihood maximization.

We provide experimental evidence on several real-data comparing manifold optimization to EM. As may be gleaned from our results, manifold optimization performs well across a wide range of parameter values and problem sizes, while being much less sensitive to overlapping data than EM, and while displaying less variability in running times.

We review key concepts of first-order deterministic manifold optimization. We also include the design and specific implementation choices of our line-search procedure. These choices ensure convergence, and are instrumental to making our Riemannian LBFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) solver outperform both EM and Riemannian Conjugate Gradients (CG). This solver should be of independent interest too.

We will also release a Matlab implementation of the methods developed in this paper. The manifold CG method that we use is directly based on the excellent toolkit ManOpt [9].

1.1 Related work

EM is such a widely studied method, that we have no hope of summarizing all the related work, even if we restrict to just Gmms. Let us instead mention a few lines of related work. Xu and Jordan [44] examine several aspects of EM for Gmms and counter the claims of Redner and Walker [32], who thought EM to be inferior to general purpose nonlinear programming methods, especially second-order methods. However, it is well-known (see e.g., [32,44]) that EM can attain good likelihood values rapidly, and that it scales to larger problems than amenable to second-order methods. Local convergence analysis of EM is available in [44], with more refined and precise results in [26], who formally show that when data have low overlap, EM can converge locally superlinearly. Our paper uses manifold LBFGS, which being a quasi-Newton method can display local superlinear convergence, though this capability is not the focus of our paper. Recently, Balakrishnan et al. [4] showed initializing closed enough to the global minimum, EM algorithm converges to the global minimum of the cost function. Their analysis aims to quantify a Euclidean ball around the global optimum by assuming a large-enough signal to noise ratio and also assuming large-enough sample sizes. Since local and global optimality are topological properties of the cost function, independent of the geometry imposed on the parameters (Euclidean, Riemannian, or otherwise), under the same assumptions as theirs one can likely construct a Riemannian ball around the global optimum, and perform a similar analysis. These aspects are of statistical value, but lie outside the scope of the present paper.

Parameter fitting using gradient-based methods has also been suggested [30,36]. Here, to satisfy positive definiteness, the authors suggest using Cholesky decompositions. These works report results only for low-dimensional problems and spherical (near spherical) covariance matrices.

Beyond EM, there is also substantial work on theoretical analysis of Gmms [4,12, 16,28]. These studies are theoretically valuable (though sometimes limited to either low-dimensional, or small number of mixture components, or spherical Gaussians, etc.), but orthogonal to our work that focuses on practical numerical algorithms for general Gmms.

The use of Riemannian optimization for Gmm is relatively new, even though manifold optimization is by now a fairly well-developed branch of optimization. A classic reference is [39]; a more recent work is [1]; and even a Matlab toolbox exists now [9]. In machine learning, manifold optimization has witnessed increasing interest[1], e.g., for low-rank optimization [23,41], optimization based on geodesic convexity [37,42], or for neural network training [43].

Beyond the choice of Riemannian metric on positive definite matrices used in this paper, one may construct other Riemannian metrics as in [2] that incorporate both the geometry of the open set and also the cost function. Afterwards, simple gradient

[1] Not to be confused with "manifold learning" a separate problem altogether.

descent method would suffice because the choice of metric includes both curvature information of the objective function and also the geometry of the open set. Beyond this, an infinite variety of other Riemannian metrics can be placed on the set of positive definite matrices. The specific choice of metric used in this paper is motivated by a variety of reasons: (i) the empirical results it leads to (as shown in the current paper); (ii) its "affine invariance" properties; (iii) the geodesic convexity that it yields, including for the single component Gaussian maximum likelihood problem, (iv) its closed form exponential and inverse exponential maps, as well as vector transport; and somewhat more abstractly, (v) its interpretation as a Fisher–Rao metric [18,19] (see also the latter references for other collections of interesting Riemannian metrics on positive definite matrices).

Bonnabel [8] studies SGD on Riemannian manifolds and focuses on its asymptotic convergence under assumptions analogous to Euclidean SGD. Only very recently, Zhang and Sra [45] developed non-asymptotic analysis (i.e., global iteration complexity analysis) for Riemannian SGD, albeit limited to the geodesically convex case. Zhang et al. [46] provide an extension to finite-sum problems via the idea of variance reduced stochastic methods (to be pedantic, randomized incremental gradient), obtaining non-asymptotic convergence analysis—due to the finite-sum assumption, their analysis is not applicable to stochastic optimization problems where one must minimize an expectation over possibly infinitely many points.

More importantly, there is a critical difference between both [45,46] and the present paper. These other works assume that the iterates generated by their stochastic procedures stay within a compact set. This assumption is crucial for their convergence analysis, and it is enforced via an additional metric projection onto a compact constraint set (which must be somehow determined a priori). In contrast, we present an analysis for Riemannian-SGD applied to GMM optimization, for which we show that the iterates remain within a compact set (see Theorem 6).

2 Background on manifold optimization

Manifolds are spaces that locally resemble a Euclidean space, and smooth manifolds have smooth transitions between locally Euclidean-like subsets [25]. The tangent space $T_x\mathcal{M}$ is an approximating vector space at each point x of the manifold $\mathcal{M}$. The tangent bundle of a smooth manifold $\mathcal{M}$ is a manifold $T\mathcal{M}$, which assembles all the tangents in that manifold, $T\mathcal{M} = \bigsqcup_{x\in\mathcal{M}} T_x\mathcal{M} = \{(x, y) | x \in \mathcal{M}, y \in T_x\mathcal{M}\}$. If a smooth manifold is equipped with a smoothly-varying inner product on each of its tangent spaces, it is called *Riemannian manifold*.

This additional structure of a Riemannian manifold proves very useful in developing optimization techniques specific to manifolds [39]. Indeed, it is easy to extend unconstrained optimization techniques to smooth manifolds, at least from the perspective of asymptotic complexity analysis [1]; though the non-asymptotic case is considerably more complicated [45,46].

The key manifold in this paper is $\mathbb{P}^d$, the manifold of $d \times d$ symmetric positive definite (PSD) matrices. At a point $\boldsymbol{\Sigma} \in \mathbb{P}^d$, the tangent space $T_{\boldsymbol{\Sigma}}\mathbb{P}^d$ is isomorphic to the entire set of symmetric matrices; and the Riemannian metric at $\boldsymbol{\Sigma}$ between two vectors ξ and η in $T_{\boldsymbol{\Sigma}}\mathbb{P}^d$ is given by

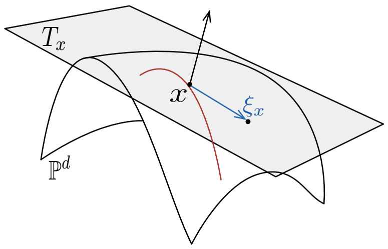

Fig. 1 Visualization of line-search on a manifold: x is a point on the manifold, T_x is the tangent space at the point x, ξ_x is a descent direction at x; the red curve is the curve along which line-search is performed

$$g_{\boldsymbol{\Sigma}}(\xi, \eta) := \operatorname{tr}(\boldsymbol{\Sigma}^{-1}\xi\boldsymbol{\Sigma}^{-1}\eta).$$

Riemannian manifolds have *geodesics*, which are curves that (locally) join points along shortest paths which depends on the choice of Riemannian metric. Geodesics help generalize the notion of convexity to manifolds.

2.1 Geodesic convexity

Let $\mathcal{M}$ be a Riemannian manifold and γ_{xy} a geodesic from x to y; that is

$$\gamma_{xy} : [0, 1] \to \mathcal{M}, \quad \gamma_{xy}(0) = x,\ \gamma_{xy}(1) = y.$$

A set $\mathcal{A} \subseteq \mathcal{M}$ is *geodesically convex* (henceforth g-convex) if for all $x, y \in \mathcal{A}$ there is a geodesic γ_{xy} contained within $\mathcal{A}$. Further, a function $f : \mathcal{A} \to \mathbb{R}$ is g-convex if for all $x, y \in \mathcal{A}$, the composition $f \circ \gamma_{xy} : [0, 1] \to \mathbb{R}$ is convex in the usual Euclidean sense.

The Riemannian metric on $\mathbb{P}^d$ mentioned above induces a geodesic between two points $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ that has the well-known closed-form (see e.g., [5, Ch. 6]):

$$\gamma_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(t) := \boldsymbol{\Sigma}_1^{1/2}\left(\boldsymbol{\Sigma}_1^{-1/2}\boldsymbol{\Sigma}_2\boldsymbol{\Sigma}_1^{-1/2}\right)^t\boldsymbol{\Sigma}_1^{1/2}, \quad 0 \le t \le 1.$$

Thus, a function $f : \mathbb{P}^d \to \mathbb{R}$ if g-convex on $\mathbb{P}^d$ if it satisfies

$$f(\gamma_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(t)) \le (1-t)f(\boldsymbol{\Sigma}_1) + tf(\boldsymbol{\Sigma}_2), \qquad t \in [0, 1],\ \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2 \in \mathbb{P}^d.$$

The negative of a g-convex function is called g-concave. For a g-convex function, local optimality implies global optimality even if it is nonconvex in the Euclidean case. This remarkable property follows easily from g-convexity upon mimicking the corresponding Euclidean proof. This property has been investigated in some matrix theoretic applications [5,38], and has been used in recent theoretical and applied works in nonlinear optimization [35,37,42,45].

2.2 First-order methods for Riemannian optimization

At a high-level, first-order methods for manifold optimization methods operate iteratively as follows (see Fig. 1 for a conceptual demonstration):

(i) Obtain a descent direction, namely, a vector in tangent space that decreases the cost function if we infinitesimally move along it;
(ii) Perform a line-search along a smooth curve on the manifold to obtain sufficient decrease and ensure convergence.

Such a smooth curve that is parametrized by a point on the manifold and a (descent) direction is called retraction. A retraction is a smooth mapping Ret from the tangent bundle $T\mathcal{M}$ to the manifold $\mathcal{M}$. The restriction of retraction to $T_x\mathcal{M}$, $\mathrm{Ret}_x : T_x\mathcal{M} \to \mathcal{M}$, is a smooth mapping with

(1) $\mathrm{Ret}_x(0) = x$, where 0 denotes the zero element of $T_x\mathcal{M}$.
(2) $D\,\mathrm{Ret}_x(0) = \mathrm{id}_{T_x\mathcal{M}}$, where $D\,\mathrm{Ret}_x$ denotes the derivative of Ret_x and $\mathrm{id}_{T_x\mathcal{M}}$ denotes the identity mapping on $T_x\mathcal{M}$.

One possible candidate for retraction on Riemannian manifolds is the exponential map. The exponential map $\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M}$ is defined as $\mathrm{Exp}_x\, v = \gamma(1)$, where γ is the geodesic satisfying the conditions $\gamma(0) = x$ and $\dot{\gamma}(0) = v$. The reader is referred to [1,39] for more in depth discussion.

First-order methods are based on gradients. The gradient on a Riemannian manifold is defined as the vector $\nabla f(x)$ in tangent space such that

$$Df(x)\xi = \langle \nabla f(x), \xi \rangle, \quad \text{for } \xi \in T_x\mathcal{M},$$

where $\langle \cdot, \cdot \rangle$ is the inner product in the tangent space $T_x\mathcal{M}$. $Df(x)\xi$ is the directional derivative of f along ξ. Let $\gamma : [-1, 1] \to \mathcal{M}$ be a differentiable curve with $\gamma(0) = x$ and $\gamma'(0) = \xi$, then the directional derivative can be defined by

$$Df(x)\xi = \frac{d}{dt} f \circ \gamma(\tau)\bigg|_{\tau=0}.$$

Differentials at each point on the manifold forms the cotangent space. The cotangent space on the smooth manifold $\mathcal{M}$ at point x is defined as the dual space of the tangent space. Elements of the cotangent space are linear functionals on the tangent space.

Another important concept needed for methods like conjugate-gradient and LBFGS is *vector transport*. Vector transport is a smooth function that allows moving tangent vectors along retractions. A vector transport $\mathcal{T} : \mathcal{M} \times \mathcal{M} \times T\mathcal{M} \to T\mathcal{M}$, $(x, y, \xi) \mapsto \mathcal{T}_{x,y}(\xi)$ is a mapping satisfying the following properties:

(1) There exists an associated retraction Ret and a tangent vector ν satisfying $\mathcal{T}_{x,y}(\xi) \in T_{\mathrm{Ret}_x(\nu)}$, for all $\xi \in T_x\mathcal{M}$.
(2) $\mathcal{T}_{x,y}\xi = \xi$, for all $\xi \in T_x\mathcal{M}$.
(3) The mapping $\mathcal{T}_{x,y}(.)$ is linear.

An important special case of vector transport is parallel transport, which is defined as a differential map between tangent spaces at different points on the manifold with zero derivative along a smooth curve connecting the points. The differential map between tangent spaces on the manifold is a smooth vector field, where a vector field is an assignment of a tangent vector to each point on a manifold. For computing the derivative of such a map, one first needs to define a connection, which is a way

Table 1 Summary of Riemannian expressions for PSD matrices

Definition	Expression for the PSD manifold
Tangent space	Space of symmetric matrices
Metric between ξ, η at Σ	$g_\Sigma(\xi, \eta) = \operatorname{tr}(\Sigma^{-1}\xi\Sigma^{-1}\eta)$
Gradient at Σ if Euclidean gradient is $\nabla_E f$	$\nabla f(\Sigma) = \frac{1}{2}\Sigma(\nabla_E f(\Sigma) + [\nabla_E f(\Sigma)]^T)\Sigma$
Exponential map at Σ in direction ξ	$\operatorname{Exp}_\Sigma(\xi) = \Sigma \exp(\Sigma^{-1}\xi)$
Parallel transport of ξ from Σ_1 to Σ_2	$\mathcal{T}_{\Sigma_1,\Sigma_2}(\xi) = E\xi E^T, \quad E = (\Sigma_2\Sigma_1^{-1})^{1/2}$
Euclidean Retraction at Σ in direction ξ	$\operatorname{Ret}_\Sigma(\xi) = \Sigma + \xi$

to perform directional derivative of vector fields. Let $\mathcal{V}(\mathcal{M})$ be the set of smooth vector fields on $\mathcal{M}$, a connection is a map $\nabla : \mathcal{V}(\mathcal{M}) \times \mathcal{V}(\mathcal{M}) \to \mathcal{V}(\mathcal{M})$ satisfying certain properties [1]. Given a smooth curve $\gamma : [0, 1] \to \mathcal{M}$, transporting a vector $\nu_0 \in T_{\gamma(0)}$ to a vector $\nu(t) \in T_{\gamma(t)}$ can be done by solving the following initial value problem

$$\nabla_{\dot\gamma(t)}\nu = 0, \quad \nu(0) = \nu_0.$$

For $x = \gamma(0)$ and $y = \gamma(t)$, the parallel transport of $\nu_0 \in T_x\mathcal{M}$ to $\nu(t) \in T_y\mathcal{M}$ is a vector transport $\nu(t) = T_{x,y}\nu_0$.

Table 1 summarizes the key quantities for $\mathbb{P}^d$. The Euclidean retraction is not positivity-preserving, meaning that for large ξ the retraction may not stay in the space $\mathbb{P}^d$, and is therefore not valid. Because of its non positivity-preserving behavior, the Euclidean retraction can not be used easily in line-search algorithms and therefore it is not used in our batch optimization algorithms.

If the parameter space is a product space of several manifolds, the concepts can be easily defined based on individual manifolds. For example, the exponential map, gradient and parallel transport are defined as the Cartesian product of individual expressions, and the inner product is defined as the sum of inner product of the components in their respective manifolds.

2.3 Stochastic optimization

If the objective function has the form

$$\min_{X\in\mathcal{M}} \quad f(x) := \frac{1}{n}\sum\nolimits_{i=1}^n f_i(x), \tag{3}$$

then for large n each iteration of the first-order methods explained above becomes very expensive, as merely computing the gradient requires going through all n component functions. In this large-scale setting, one frequently passes to stochastic/incremental optimization methods such as stochastic gradient descent (SGD) that processes only a small batch of functions at each iteration. Note that SGD is actually not a descent method; it makes progress by replacing an exact descent direction by one which is a descent direction in expectation.

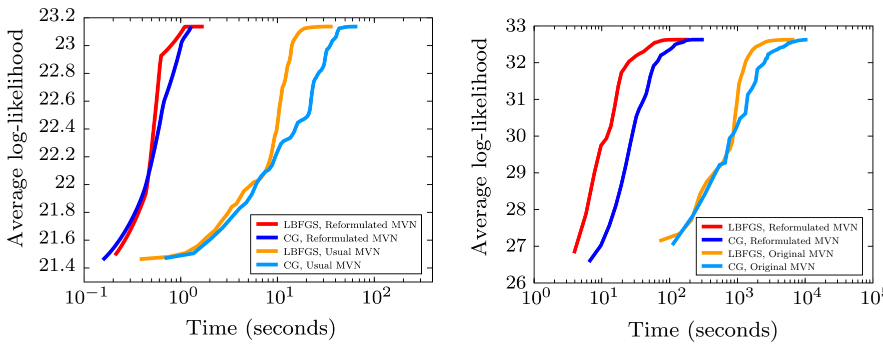

Fig. 2 The effect of reformulation in convergence speed of manifold CG and manifold LBFGS methods ($d = 35$); note that the X-axis (time) is on a logarithmic scale [20]

Riemannian SGD [8] runs the following iteration, where $i_t \sim U(n)$, i.e. a random integer between 1 and n:

$$x_{t+1} \leftarrow \mathrm{Ret}_{x_t}(-\eta_t \nabla f_{i_t}(x_t)), \qquad t = 0, 1, \ldots, \tag{4}$$

where Ret_x is a retraction at the point x and η_t is a suitable stepsize that typically satisfies $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$.

After this background on the Riemannian optimization methods that we will use for GMM parameter optimization, we are now ready to describe the problem reformulation and other important theoretical details.

3 Problem reformulation

Experience with mixture modeling shows that whenever an optimization method works well for a single component, the same optimization method also works well for the mixture model. We begin, therefore, with parameter estimation for a single Gaussian. Although this problem has a closed-form solution that benefits EM, our goal is to tackle it in the context of manifold optimization.

Consider, maximum likelihood parameter estimation for a single Gaussian,

$$\max_{\boldsymbol{\mu}, \boldsymbol{\Sigma} \succ 0} \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) := \sum_{i=1}^{n} \log p_{\mathcal{N}}(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{5}$$

This objective is concave in the Euclidean sense. But our aim is to apply manifold optimization and this objective is *not* g-concave on its domain $\mathbb{R}^d \times \mathbb{P}^d$, which makes it geometrically somewhat of a mismatch.

We invoke a simple transformation that turns (5) into a g-concave optimization problem. This transformation has a dramatic impact on the speed of the convergence for a single Gaussian, as seen in Fig. 2. Define new vectors $\boldsymbol{y}_i^T = [\boldsymbol{x}_i^T \; 1]$; then, the proposed transformed model is

$$\max_{\boldsymbol{S}\succ 0} \ \widehat{\mathcal{L}}(\boldsymbol{S}) := \sum_{i=1}^{n} \log q_{\mathcal{N}}(\boldsymbol{y}_i; \boldsymbol{S}), \tag{6}$$

where $q_{\mathcal{N}}(\boldsymbol{y}_i; \boldsymbol{S}) := 2\pi \exp(\frac{1}{2}) p_{\mathcal{N}}(\boldsymbol{y}_i; \boldsymbol{S})$. Note that this new cost function is *not* just a reparametrization of (5). However, it becomes a reparametrization at a maximum. More precisely, Theorem 1 shows that solving the reformulation (6) also solves the original problem (5).

Theorem 1 *If* $\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*$ *maximize* (5)*, and if* $\boldsymbol{S}^*$ *maximizes* (6)*, then* $\widehat{\mathcal{L}}(\boldsymbol{S}^*) = \mathcal{L}(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ *and*

$$\boldsymbol{S}^* = \begin{pmatrix} \boldsymbol{\Sigma}^* + \boldsymbol{\mu}^*\boldsymbol{\mu}^{*T} & \boldsymbol{\mu}^* \\ \boldsymbol{\mu}^{*T} & 1 \end{pmatrix}. \tag{7}$$

Proof We express $\boldsymbol{S}$ by new variables $\boldsymbol{U}$, $\boldsymbol{t}$ and s by writing

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{U} + s\boldsymbol{t}\boldsymbol{t}^T & s\boldsymbol{t} \\ s\boldsymbol{t}^T & s \end{pmatrix}. \tag{8}$$

The objective function $\widehat{\mathcal{L}}(\boldsymbol{S})$ in terms of the new parameters becomes

$$\begin{aligned}\widehat{\mathcal{L}}(\boldsymbol{U}, \boldsymbol{t}, s) &= \tfrac{n}{2} - \tfrac{d}{2}\log(2\pi) - \tfrac{n}{2}\log s - \tfrac{n}{2}\log\det(\boldsymbol{U}) \\ &\quad - \sum_{i=1}^{n} \tfrac{1}{2}(\boldsymbol{x}_i - \boldsymbol{t})^T \boldsymbol{U}^{-1}(\boldsymbol{x}_i - \boldsymbol{t}) - \tfrac{n}{2s}.\end{aligned}$$

Optimizing $\widehat{\mathcal{L}}$ over $s > 0$ we see that $s^* = 1$. Hence, the objective reduces to a d-dimensional Gaussian log-likelihood, for which $\boldsymbol{U}^* = \boldsymbol{\Sigma}^*$ and $\boldsymbol{t}^* = \boldsymbol{\mu}^*$. □

In other words, Theorem 1 shows that our model transformation is "faithful" because it leaves the optimum unchanged. Figure 2 shows the unmistakable impact this transformation has on the convergence speed of Riemannian CG and Riemannian LBFGS.

Next, Proposition 1 proves another key property of this transformation: the objective in (6) becomes g-concave. For proving Proposition 1, we need the following lemma that is an easy consequence of [5, Thm. 4.1.3]:

Lemma 1 *Let* $\boldsymbol{S}, \boldsymbol{R} \succ 0$*. Then, for a vector* $\boldsymbol{x}$ *of appropriate dimension,*

$$\boldsymbol{x}^T(\boldsymbol{S}^{-1/2}(\boldsymbol{S}^{1/2}\boldsymbol{R}^{-1}\boldsymbol{S}^{1/2})^{1/2}\boldsymbol{S}^{-1/2})\boldsymbol{x} \le [\boldsymbol{x}^T\boldsymbol{S}^{-1}\boldsymbol{x}]^{1/2}[\boldsymbol{x}^T\boldsymbol{R}^{-1}\boldsymbol{x}]^{1/2}. \tag{9}$$

Proposition 1 *The objective* $\widehat{\mathcal{L}}(\boldsymbol{S})$ *in* (6) *is g-concave.*

Proof By definition, for proving geodesic concavity, we need to prove that $\widehat{\mathcal{L}}(\gamma_{\boldsymbol{S},\boldsymbol{R}}(.))$ is concave in the usual Euclidean sense. Because this function is continuous, it suffices to establish mid-point geodesic concavity:

$$\widehat{\mathcal{L}}(\gamma_{\boldsymbol{S},\boldsymbol{R}}(\tfrac{1}{2})) \ge \tfrac{1}{2}\widehat{\mathcal{L}}(\boldsymbol{S}) + \tfrac{1}{2}\widehat{\mathcal{L}}(\boldsymbol{R}), \qquad \text{for } \boldsymbol{S}, \boldsymbol{R} \in \mathbb{P}^d.$$

Denoting inessential constants by c, the above inequality turns into

$$\begin{aligned}\widehat{\mathcal{L}}(\gamma_{\boldsymbol{S},\boldsymbol{R}}(\tfrac{1}{2})) &= -\log\det(\boldsymbol{S}^{1/2}\boldsymbol{R}^{1/2}) - c\sum_i \boldsymbol{y}_i^T(\boldsymbol{S}^{-1/2}(\boldsymbol{S}^{1/2}\boldsymbol{R}^{-1}\boldsymbol{S}^{1/2})^{1/2}\boldsymbol{S}^{-1/2})\boldsymbol{y}_i \\ &\geq -\tfrac{1}{2}\log\det(\boldsymbol{S}) - \tfrac{1}{2}\log\det(\boldsymbol{R}) - c\sum_i [\boldsymbol{y}_i^T\boldsymbol{S}^{-1}\boldsymbol{y}_i]^{1/2}[\boldsymbol{y}_i^T\boldsymbol{R}^{-1}\boldsymbol{y}_i]^{1/2} \\ &\geq -\tfrac{1}{2}\log\det(\boldsymbol{S}) - \tfrac{c}{2}\sum_i \boldsymbol{y}_i^T\boldsymbol{S}^{-1}\boldsymbol{y}_i - \tfrac{1}{2}\log\det(\boldsymbol{R}) - \tfrac{c}{2}\sum_i \boldsymbol{y}_i^T\boldsymbol{R}^{-1}\boldsymbol{y}_i \\ &= \tfrac{1}{2}\widehat{\mathcal{L}}(\boldsymbol{S}) + \tfrac{1}{2}\widehat{\mathcal{L}}(\boldsymbol{R}),\end{aligned}$$

where the first inequality is follows from Lemma 1. □

Theorem 2 *A local maximum of the reformulated* GMM *log-likelihood*

$$\widehat{\mathcal{L}}\left(\{\boldsymbol{S}_j\}_{j=1}^K\right) := \sum_{i=1}^n \log\left(\sum_{j=1}^K \alpha_j q_{\mathcal{N}}(\boldsymbol{y}_i;\boldsymbol{S}_j)\right)$$

is a local maximum of the original log-likelihood

$$\mathcal{L}\left(\{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j\}_{j=1}^K\right) := \sum_{i=1}^n \log\left(\sum_{j=1}^K \alpha_j p_{\mathcal{N}}(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)\right).$$

Proof Let $\boldsymbol{S}_1^*,\dots,\boldsymbol{S}_K^*$ be a local maximum of $\widehat{\mathcal{L}}$. Then, $\boldsymbol{S}_j^*$ is the maximum of the following cost function:

$$-\frac{1}{2}\sum\nolimits_{i=1}^n w_i \log\det(\boldsymbol{S}_j) - \frac{1}{2}\sum_{i=1}^n w_i \boldsymbol{y}_i^T\boldsymbol{S}_j^{-1}\boldsymbol{y}_i,$$

where for each $i \in \{1,\dots,n\}$ the weight

$$w_i = \frac{q_{\mathcal{N}}(\boldsymbol{y}_i|\boldsymbol{S}_j^*)}{\sum_{j=1}^K \alpha_j q_{\mathcal{N}}(\boldsymbol{y}_i|\boldsymbol{S}_j^*)}. \tag{10}$$

Using an argument similar to that for Theorem 1, we see that $s_j^* = 1$, whereby $q_{\mathcal{N}}(\boldsymbol{y}_i|\boldsymbol{S}_j^*) = p_{\mathcal{N}}(\boldsymbol{x}_i;\boldsymbol{t}_j^*,\boldsymbol{U}_j^*)$. Thus, at a maximum the objective functions agree and the proof is complete. □

Theorem 2 shows that we can replace (2) by a reformulated log-likelihood whose local maxima agree with those of (2). Moreover, the individual components of the reformulated log-likelihood are geodesically concave.

Finally, we also need to replace the constraint $\boldsymbol{\alpha} \in \Delta_K$ to make the problem unconstrained. We do this via a commonly used change of variables [22]:

$$\omega_k = \log\left(\frac{\alpha_k}{\alpha_K}\right), \quad k = 1, \ldots, K-1. \tag{11}$$

Assume $\omega_K = 0$ to be a constant; then the final optimization problem is:

$$\max_{\{S_j \succ 0\}_{j=1}^{K}, \{\omega_j\}_{j=1}^{K-1}} \widehat{\mathcal{L}}\big(\{S_j\}_{j=1}^{K}, \{\omega_j\}_{j=1}^{K-1}\big) := \sum_{i=1}^{n} \log\left(\sum_{j=1}^{K} \frac{\exp(\omega_j)}{\sum_{k=1}^{K}\exp(\omega_k)} q_{\mathcal{N}}(\boldsymbol{y}_i; \boldsymbol{S}_j)\right) \tag{12}$$

We solve (12) via Riemannian optimization problem in this paper; specifically, it is an optimization problem on the product manifold $\big(\prod_{j=1}^{K} \mathbb{P}^{d+1}\big) \times \mathbb{R}^{K-1}$.

3.1 Formulations for penalized likelihoods

One of the problems with ML estimation for GMMs is covariance singularity. There are several remedies to avoid this problem, and the most common approach is to use a penalized ML estimate [34]. In our reformulation (6), to avoid problems due to singularity, a suitable regularizer $\psi(\boldsymbol{S})$ needs to be added to the objective function. We state the following generic results that helps us choose priors amenable to our framework.

Theorem 3 *Let $\boldsymbol{S}$ be the block matrix defined in* (8). *Consider a regularizer that splits over the blocks of $\boldsymbol{S}$, and has the form*

$$\psi(\boldsymbol{S}) = \psi_1(\boldsymbol{U}, \boldsymbol{t}) + \psi_2(s),$$

where $\psi_2(s)$ has a unique maximizer at $s = 1$. Let $\boldsymbol{S}^*$ be the maximum of the penalized objective $\psi(\boldsymbol{S}) + \widehat{\mathcal{L}}(\boldsymbol{S})$, where $\widehat{\mathcal{L}}(\boldsymbol{S})$ is the modified log-likelihood (6). *Assume that $(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ maximizes the penalized log-likelihood $\psi_1(\boldsymbol{\Sigma}, \boldsymbol{\mu}) + \mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is as in* (5). *Then, $\boldsymbol{S}^*$ is related to $(\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)$ via* (7).

Proof Similar to the proof of Theorem 1, it is easy to see that the penalized objective $\psi + \widehat{\mathcal{L}}$ has its maximum at $s^* = 1$. Therefore, the objective reduces to a penalized log-likelihood of a Gaussian at its maximum. □

A widely used penalizer is obtained by placing an inverse Wishart prior on covariance matrices and using a *maximum* a priori estimate. The inverse Wishart prior is a conjugate prior for the covariance matrix, and is given by

$$p(\boldsymbol{\Sigma}; \boldsymbol{\Lambda}; \nu) \propto \det(\boldsymbol{\Sigma})^{-(\nu+d+1)/2} \exp\big(-\tfrac{1}{2}\operatorname{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Lambda})\big),$$

where ν is a degree of freedom and $\boldsymbol{\Lambda}$ is a scale parameter. The conjugate prior for the mean parameter is a Gaussian distribution conditioned on the covariance matrix; that is,

$$p(\boldsymbol{\mu}|\boldsymbol{\Sigma}; \boldsymbol{\lambda}, \kappa) \propto \det(\boldsymbol{\Sigma})^{-1/2} \exp\big(-\tfrac{\kappa}{2}(\boldsymbol{\mu} - \boldsymbol{\lambda})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \boldsymbol{\lambda})\big),$$

where κ is a so-called shrinkage parameter.

In the following, we propose a penalizer to our reformulated objective function. This penalized objective function converges to the penalized log-likelihood for GMM, when one uses the aforementioned conjugate priors for covariance matrices and means. Consider the penalizer

$$\psi(\boldsymbol{S}; \boldsymbol{\Psi}) = -\frac{\rho}{2} \log \det(\boldsymbol{S}) - \beta \tfrac{1}{2} \operatorname{tr}(\boldsymbol{\Psi} \boldsymbol{S}^{-1}), \tag{13}$$

where $\boldsymbol{\Psi}$ is the block matrix

$$\boldsymbol{\Psi} = \begin{pmatrix} \frac{\alpha}{\beta}\boldsymbol{\Lambda} + \kappa\boldsymbol{\lambda}\boldsymbol{\lambda}^T & \kappa\boldsymbol{\lambda} \\ \kappa\boldsymbol{\lambda}^T & \kappa \end{pmatrix}, \tag{14}$$

and the parameter $\rho = \alpha(d + \nu + 1) + \beta$. By construction $\boldsymbol{\Psi}$ is a positive definite matrix. The matrix $\boldsymbol{\Lambda}$ is positive definite and therefore the first block $\frac{\alpha}{\beta}\boldsymbol{\Lambda} + \kappa\boldsymbol{\lambda}\boldsymbol{\lambda}^T$ is also positive definite. Therefore, for showing positive definiteness of $\boldsymbol{\Psi}$ it suffices to show that the determinant of $\boldsymbol{\Psi}$ is positive. This is true because the determinant is equal to $\det\big(\frac{\alpha}{\beta}\boldsymbol{\Lambda}\big)\kappa$.

If we write $\boldsymbol{S}$ as the block matrix

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{U} + s\boldsymbol{t}\boldsymbol{t}^T & s\boldsymbol{t} \\ s\boldsymbol{t}^T & s \end{pmatrix},$$

then the penalized cost function (13) becomes

$$\begin{aligned} \psi(\boldsymbol{S}; \boldsymbol{\Psi}) &= -\frac{\rho}{2}\big[\log\det(\boldsymbol{U}) + \log(s)\big] \\ &\quad - \tfrac{\beta}{2}\left[\tfrac{\alpha}{\beta}\operatorname{tr}(\boldsymbol{\Lambda}\boldsymbol{U}^{-1}) + \kappa(\boldsymbol{t}^T\boldsymbol{U}^{-1}\boldsymbol{t}) + \kappa(\boldsymbol{\lambda}^T\boldsymbol{U}^{-1}\boldsymbol{\lambda}) - 2\kappa\boldsymbol{\lambda}^T\boldsymbol{U}^{-1}\boldsymbol{t} + \tfrac{\kappa}{s}\right]. \end{aligned}$$

Rearranging the terms, we thus obtain

$$\psi(\boldsymbol{S}; \boldsymbol{\Psi}) = \alpha \log p(\boldsymbol{U}; \boldsymbol{\Lambda}; \nu) + \beta \log p(\boldsymbol{t}|\boldsymbol{U}; \boldsymbol{\lambda}, \kappa) - \frac{\rho}{2}\log(s) - \frac{\beta\kappa}{2s} + c, \tag{15}$$

for some constant c. In order for this penalizer to satisfy the conditions of Theorem 3 we need the following condition:

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

Using Proposition 1 one can again show that this penalizer is g-concave. We summarize these results as an informal corollary below.

Corollary 1 *The penalizer given in (15) is g-concave and fulfills the structure required by Theorem 3. Hence, it can be used for penalized ML estimation.*

It is easy to see that the single component results above extend to penalized maximum likelihood of GMMs. That is, Theorem 2 can be generalized to penalized maximum likelihood for GMMs.

Indeed, recall that a common prior on mixture weights is the symmetric Dirichlet prior that assumes the form

$$p(\alpha_1, \dots, \alpha_K; \zeta) \propto \prod_{i=1}^{K} \alpha_i^{\zeta}. \tag{16}$$

The penalizer for the mixture weights is the logarithm of (16), namely,

$$\varphi\left(\{\omega_j\}_{i=1}^{K-1}; \zeta\right) := \zeta \sum_{i=1}^{K} \log\left(\frac{e^{\omega_j}}{\sum_{k=1}^{K} e^{\omega_k}}\right) = \zeta \sum_{i=1}^{K} \omega_i - K\zeta \log\left(\sum_{k=1}^{K} e^{\omega_k}\right). \tag{17}$$

The final optimization problem for the penalized mixture model is

$$\max_{\{S_j \succ 0\}_{j=1}^K, \{\omega_j\}_{j=1}^{K-1}} \widehat{\mathcal{L}}\left(\{\boldsymbol{S}_j\}_{j=1}^K, \{\omega_j\}_{j=1}^{K-1}\right) + \sum_{j=1}^{K} \psi(\boldsymbol{S}_j; \boldsymbol{\Psi}) + \varphi\left(\{\omega_j\}_{i=1}^{K-1}; \zeta\right), \tag{18}$$

where $\widehat{\mathcal{L}}(\{\boldsymbol{S}_j\}_{j=1}^K, \{\omega_j\}_{j=1}^{K-1})$, $\psi(\boldsymbol{S}; \boldsymbol{\Psi})$ and $\varphi(\{\omega_j\}_{i=1}^{K-1}; \zeta)$ are given by (12), (13), and (17), respectively.

We have now presented our formulation of the main optimization problems of this paper, both GMM fitting, as well as a penalized version based on using conjugate priors on means and covariance matrices combined with a Dirichlet model for mixture components weights. We can solve both these problems using Riemannian LBFGS procedure or a Riemannian SGD method for larger scale problems. The former method was also studied in [20]; we thus dedicate Sect. 4 to an general analysis Riemannian SGD before specializing it to our GMM problems in Sect. 5.

4 Riemannian stochastic optimization

In this section, we consider the stochastic gradient descent algorithm

$$x_{t+1} \leftarrow \mathrm{Ret}_{x_t}(-\eta_t \nabla f_{i_t}(x_t)), \qquad t = 0, 1, \dots, \tag{19}$$

where Ret_x is a suitable retraction (to be specialized later). We assume for our analysis of (19) the following fairly standard conditions:

(i) The function satisfies the Lipschitz growth bound

$$f(\mathrm{Ret}_x(\xi)) \le f(x) + \langle \nabla f(x), \xi \rangle + \tfrac{L}{2}\|\xi\|^2. \tag{20}$$

(ii) The stochastic gradients in all iterations are unbiased, i.e.,

$$\mathbb{E}[\nabla f_{i_t}(x_t) - \nabla f(x_t)] = 0.$$

(iii) The stochastic gradients have bounded variance, so that

$$\mathbb{E}[\|\nabla f_{i_t}(x_t) - \nabla f(x_t)\|^2] \le \sigma^2, \qquad 0 \le \sigma < \infty.$$

When the retraction is the exponential map, condition (i) can be reexpressed as (provided that $\mathrm{Exp}_y^{-1}(\cdot)$ exists)

$$f(x) - f(y) - \langle \nabla f(y), \mathrm{Exp}_y^{-1}(x)\rangle \le \tfrac{L}{2} d^2(x, y). \tag{21}$$

Given these conditions, the iterates produced by (19) satisfy the following:

Lemma 2 *Assume conditions (i)-(iii) hold. Then, the gradients in SGD satisfy the bound*

$$\sum_{t=1}^{T} \left(\eta_t^2 - \tfrac{L}{2}\eta_t^2\right) \mathbb{E}[\|\nabla f(x_t)\|^2] \le f(x_1) - f^* + \tfrac{L\sigma^2}{2} \sum_{t=1}^{T} \eta_t^2. \tag{22}$$

Proof Denote the stochastic error by $\delta_t = \nabla f(x_t) - \nabla f_{i_t}(x_t)$; also, as a shorthand set $g_t = \nabla f_{i_t}(x_t)$. Then, we have

$$\begin{aligned} f(x_{t+1}) &\le f(x_t) + \langle \nabla f(x_t), -\eta_t \nabla f_{i_t}(x_t)\rangle + \tfrac{L}{2}\|\eta_t \nabla f_{i_t}(x_t)\|^2 \\ &= f(x_t) - \eta_t \langle \nabla f(x_t), g_t\rangle + \tfrac{L\eta_t^2}{2}\|g_t\|^2 \\ &= f(x_t) - \eta_t \|\nabla f(x_t)\|^2 - \eta_t \langle \nabla f(x_t), \delta_t\rangle \\ &\quad + \tfrac{L\eta_t^2}{2}\left[\|\nabla f(x_t)\|^2 + 2\langle \nabla f(x_t), \delta_t\rangle + \|\delta_t\|^2\right] \\ &= f(x_t) - \left(\eta_t^2 - \tfrac{L}{2}\eta_t^2\right)\|\nabla f(x_t)\|^2 - \left(\eta_t - L\eta_t^2\right)\langle \nabla f(x_t), \delta_t\rangle + \tfrac{L\eta_t^2}{2}\|\delta_t\|^2. \end{aligned}$$

Summing over $t = 1, \ldots, T$, using telescoping sums and rearranging we obtain

$$\begin{aligned} &\sum_{t=1}^{T} \left(\eta_t^2 - \tfrac{L}{2}\eta_t^2\right)\|\nabla f(x_t)\|^2 \\ &\quad \le f(x_1) - f(x_{T+1}) - \sum_{t=1}^{T}\left(\eta_t - L\eta_t^2\right)\langle \nabla f(x_t), \delta_t\rangle + \frac{L}{2}\sum_{t=1}^{T}\eta_t^2\|\delta_t\|^2 \\ &\quad \le f(x_1) - f^* - \sum_{t=1}^{T}\left(\eta_t - L\eta_t^2\right)\langle \nabla f(x_t), \delta_t\rangle + \frac{L}{2}\sum_{t=1}^{T}\eta_t^2\|\delta_t\|^2, \end{aligned}$$

where we used $f^* \le f(x_t)$ for all t. Now taking expectations, and noting that by our assumption $\mathbb{E}[\|\delta_t\|^2] \le \sigma^2$ while by unbiasedness of the stochastic gradients we have $\mathbb{E}[\langle \nabla(x_t), \delta_t\rangle] = 0$. Thus, we obtain the bound (22). ◻

By using a specific choice of parameter η_t and using Lemma 2, we can obtain a convergence rate result for SGD with a slight modification.

Theorem 4 *Assume a slightly modified version of SGD which output a point x_a by randomly picking one of the iterates, say x_t, with probability $p_t := (2\eta_t - L\eta_t^2)/Z_T$, where $Z_T = \sum_{t=1}^{T}(2\eta_t - L\eta_t^2)$. Furthermore, choose $\eta_t = \min\{L^{-1}, c\sigma^{-1}T^{-1/2}\}$ for a suitable constant c. Then, we obtain the following bound on $\mathbb{E}[\|\nabla f(x_a)\|^2]$, which measures the expected gap to stationarity:*

$$\mathbb{E}[\|\nabla f(x_a)\|^2] \leq \frac{2L\Delta_1}{T} + \left(c + c^{-1}\Delta_1\right)\frac{L\sigma}{\sqrt{T}} = \mathcal{O}\left(\frac{1}{T}\right) + \mathcal{O}\left(\frac{1}{\sqrt{T}}\right). \tag{23}$$

Proof Using the definition of x_a and using Lemma 2, we immediately have

$$\mathbb{E}[\|\nabla f(x_a)\|^2] = \sum_{t=1}^{T} p_t \mathbb{E}[\|\nabla f(x_t)\|^2] \leq \frac{2(f(x_1) - f^*)}{Z_T} + L\sigma^2 \frac{\sum_{t=1}^{T}\eta_t^2}{Z_T}.$$

Using the choice of η_t in the theorem, this bound yields (23). □

Theorem 4 uses a randomized stopping rule, a choice motivated by [17]. If one wishes to avoid such a rule, then under a stronger assumption one can obtain the same rate. Specifically, in the theorem below we replace conditions (ii) and (iii) with the stronger condition (iv).

(iv) The function f has a G-bounded gradient, that is $\|\nabla f_i(x)\| \leq G$ for all $i \in [n]$

Under this condition, we can obtain the following convergence rate.

Theorem 5 *Assume conditions (i) and (iv) hold. Then, the gradient in SGD satisfies the following bound for a suitable choice of η_t:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla f(x_t)\|^2] \leq \frac{1}{\sqrt{T}}\left(\frac{f(x_1) - f(x_*)}{c} + \frac{Lc}{2}G^2\right). \tag{24}$$

Proof The Lipschitz smoothness condition yields

$$\begin{aligned}\mathbb{E}[f(x_{t+1})] &\leq \mathbb{E}[f(x_t)] + \mathbb{E}\Big[\langle\nabla f(x_t),\, -\eta_t\nabla f_{i_t}(x_t)\rangle + \tfrac{L}{2}\|\eta_t\nabla f_{i_t}(x_t)\|^2\Big]\\ &\leq \mathbb{E}[f(x_t)] - \eta_t\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] + \tfrac{L\eta_t^2}{2}G^2.\end{aligned}$$

Rearranging the terms above we obtain

$$\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \leq \frac{1}{\eta_t}\mathbb{E}\Big[f(x_t) - f(x_{t+1})\Big] + \frac{L\eta_t}{2}G^2.$$

Choose $\eta_t = \frac{c}{\sqrt{T}}$ for some constant c and sum over $t = 0$ to $T-1$ to obtain

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] &\le \frac{1}{\sqrt{T}c}\mathbb{E}[f(x_1) - f(x_{T+1})] + \frac{Lc}{2\sqrt{T}}G^2 \\ &\le \frac{1}{\sqrt{T}}\left(\frac{f(x_1) - f(x^*)}{c} + \frac{Lc}{2}G^2\right).\end{aligned}$$

□

By optimizing over the constant c, the following corollary is immediate.

Corollary 2 *Assume conditions (i) and (iv) hold, then for suitable η_t we have*

$$\min_{1\le t\le T}\mathbb{E}[\|\nabla f(x_t)\|^2] \le \mathcal{O}\left(\frac{1}{\sqrt{T}}\right). \tag{25}$$

5 SGD for GMM

In this section, we investigate if SGD based on retractions satisfies the conditions needed for obtaining a global rate of convergence when applied to our GMM optimization problems. Since Euclidean retraction turns out to be computationally more effective than many other retractions, we perform the analysis below for Euclidean retraction.

Recall that we are maximizing a cost of the form $\frac{1}{n}\sum_{i=1}^{n} f_i(\cdot)$ using SGD. In a concrete realization, each function f_i is set to the penalized log-likelihood for a batch of observations (data points). For simpler notation, assume that each f_i corresponds to a single observation. Thus,

$$\begin{aligned}f_i\left(\{\boldsymbol{S}_j \succ 0\}_{j=1}^{K}, \{\eta_j\}_{j=1}^{K-1}\right) &= \log\left(\sum_{j=1}^{K}\frac{\exp(\eta_j)}{\sum_{k=1}^{K}\exp(\eta_k)}q_{\mathcal{N}}(\boldsymbol{y}_i; \boldsymbol{S}_j)\right) \\ &\quad + \frac{1}{n}\left(\sum_{j=1}^{K}\psi(\boldsymbol{S}_j; \boldsymbol{\Psi}) + \varphi\left(\{\omega_j\}_{i=1}^{K-1}; \zeta\right)\right),\end{aligned} \tag{26}$$

where $q_{\mathcal{N}}$, ψ and φ are as defined by (13) and (17), respectively. Since we are maximizing, the update formula for SGD is

$$\begin{aligned}&\left\{\{\boldsymbol{S}_j \succ 0\}_{j=1}^{K}, \{\omega_j\}_{j=1}^{K-1}\right\} \leftarrow \\ &\qquad \mathrm{Ret}_{\{\boldsymbol{S}_j \succ 0\}_{j=1}^{K}, \{\eta_j\}_{j=1}^{K-1}}\left(\eta_t\nabla f_i\left(\{\boldsymbol{S}_j \succ 0\}_{j=1}^{K}, \{\omega_j\}_{j=1}^{K-1}\right)\right),\end{aligned} \tag{27}$$

where i is a randomly chosen index between 1 and n.

Note that, the conditions needed for a global rate of convergence are not satisfied on the entire set of positive definite matrices. In particular, to apply our convergence results for SGD we need to show that the iterates stay within a compact set. Theorem 6 below ensures this property.

Theorem 6 *If the stepsize is smaller than one, then the iterates of SGD with Euclidean retraction for the penalized likelihood of* GMM *stay within a compact subset of the Riemannian manifold.*

Proof We write down the formula of the gradient and show that the update formula (27) guarantees that the variables remain in a compact subset. To this end, we show that the eigenvalues of matrices $\boldsymbol{S}_j$ remain bounded. Then, we show that the values of ω_j remain in a bounded set too. Then we can easily conclude that the iterations remain in a compact subset of the underlying manifold.

The Euclidean gradient of penalized log-likelihood with respect to one of the covariance matrices $\boldsymbol{S}_j$ for a single datapoint $\boldsymbol{y}_i$ is equal to

$$\nabla_E f_i(\boldsymbol{S}_j) = -\frac{w}{2}\boldsymbol{S}_j^{-1} + \frac{w}{2}\boldsymbol{S}_j^{-1}\boldsymbol{y}_i\boldsymbol{y}_i^T\boldsymbol{S}_j^{-1} - \frac{\rho}{2n}\boldsymbol{S}_j^{-1} + \frac{\beta}{2n}\boldsymbol{S}_j^{-1}\boldsymbol{\Psi}\boldsymbol{S}_j^{-1}, \tag{28}$$

where w, a weight calculated as in (10), is a positive number smaller than 1 and ρ, a small constant that appears in $\psi(\boldsymbol{S};\boldsymbol{\Psi})$, is of order of 10^{-2}. Using the update formula (27), $\boldsymbol{S}_j$ is updated by

$$\boldsymbol{S}_j \leftarrow \left(1 - \eta_t\frac{w + \rho n^{-1}}{2}\right)\boldsymbol{S}_j + \eta_t\boldsymbol{\Psi}', \tag{29}$$

where

$$\boldsymbol{\Psi}' = \frac{w}{2}\boldsymbol{y}_i\boldsymbol{y}_i^T + \frac{\beta n^{-1}}{2}\boldsymbol{\Psi}.$$

If $\eta_t \leq 1$, then the first term in (29) remains positive definite. Assume λ and λ' to be the smallest eigenvalue of $\boldsymbol{S}_j$ before and after the update of (29). Furthermore, assume the smallest eigenvalue of $\boldsymbol{S}_j$ before update be $\lambda_{\min}(\boldsymbol{S}_j) = \tau\lambda_{\min}(\boldsymbol{\Psi})$. From the update rule (29) and knowing that the smallest eigenvalue of sum of two matrices with positive eigenvalues is not smaller than sum of smallest eigenvalue of two matrices, we have

$$\lambda' \geq \lambda + \frac{\eta_t}{2}\lambda_{\min}(\boldsymbol{\Psi})\left(-\tau(w + \rho n^{-1}) + \beta n^{-1}\right).$$

If $\tau < \beta/(n+\rho)$, then $\lambda' > \lambda$. Otherwise, $\lambda' \geq \tau(1 - \frac{\eta_t}{2}(1 + \rho n^{-1}))\lambda_{\min}(\boldsymbol{\Psi}) + \frac{\eta_t}{2}\beta n^{-1}\lambda_{\min}(\boldsymbol{\Psi})$. Since $\frac{\eta_t}{2}(1 + \rho n^{-1}) < 1$, the smallest eigenvalue of $\boldsymbol{S}_j$ can not become smaller than

$$\lambda_{\min}(\boldsymbol{\Psi})\frac{\beta}{n+\rho}.$$

Now, assume λ and λ' to be the largest eigenvalue of $\boldsymbol{S}_j$ before and after the update given in (29). Furthermore, assume the largest eigenvalue of $\boldsymbol{S}_j$ before update be $\|\boldsymbol{S}_j\| = \tau\|\boldsymbol{\Psi}\|$. From the update rule (29) and knowing that the largest eigenvalue of sum of two matrices with positive eigenvalues is not larger than sum of largest eigenvalues of two matrices, we have

$$\lambda' \leq \lambda + \frac{\eta_t}{2}\|\boldsymbol{\Psi}\|\left(-\tau(w+\rho n^{-1}) + w\frac{\|\boldsymbol{y}_i\|}{\|\boldsymbol{\Psi}\|} + \beta n^{-1}\right).$$

If

$$\tau > \max_{w\in[0,1]} \frac{w\frac{\max_i\{\|\boldsymbol{y}_i\|\}}{\|\boldsymbol{\Psi}\|} + \beta n^{-1}}{w+\rho n^{-1}},$$

then $\lambda' < \lambda$. Therefore, the largest eigenvalue of $\boldsymbol{S}_j$ remains smaller than

$$\max_{w\in[0,1]} \frac{wn\max_i\{\|\boldsymbol{y}_i\|\} + \beta\|\boldsymbol{\Psi}\|}{wn+\rho}.$$

Till now, we have shown that the largest and the smallest eigenvalues of $\boldsymbol{S}_j$s remain smaller and larger than certain numbers, respectively. We use the same procedure to show that ω_js also remain in a bounded interval. The Euclidean gradient of the objective with respect to ω_j for a single data-point is given by:

$$\nabla_E f_i(\omega_j) = w - \alpha_j + \frac{\zeta}{n} - \frac{K\zeta}{n}\alpha_j.$$

If $\alpha_j < \frac{\zeta n^{-1}}{1+K\zeta n^{-1}}$, then the gradient is positive and ω_j is increased after update. From (11), it is clear that $\log(\alpha_j) \leq \omega_j$. Using the update formula $\omega_j^{\text{new}} = \omega_j + \eta_t \nabla_E f_i(\omega_j)$, we get the following lower bound:

$$\begin{aligned}
\omega_j^{\text{new}} &\geq \min_{\omega_j \geq \log\left(\frac{\zeta n^{-1}}{1+K\zeta n^{-1}}\right)} \left[\omega_j + \eta_t\left(w - \alpha_j + \frac{\zeta}{n} - \frac{K\zeta}{n}\alpha_j\right)\right] \\
&\geq \min_{\omega_j \geq \log\left(\frac{\zeta n^{-1}}{1+K\zeta n^{-1}}\right)} \left[\omega_j + \eta_t\left(1 - \exp(\omega_j) + \frac{\zeta}{n} - \frac{K\zeta}{n}\exp(\omega_j)\right)\right] \\
&= \log\left(\frac{\zeta n^{-1}}{1+K\zeta n^{-1}}\right).
\end{aligned}$$

From the definition (11), we have $\log(\alpha_i) = \omega_i - \log(\sum_{k=1}^K \exp(\omega_k))$. Using Jensen inequality, we obtain $\log(\alpha_i) \leq -\sum_{\substack{k=1\\k\neq i}}^n \omega_k$. Therefore, we obtain the following upper bound for ω_j

$$\begin{aligned}
\omega_j &\leq \log(\alpha_i) - \sum_{\substack{k=1\\k\neq i, k\neq j}}^{n} \omega_k \\
&\leq -(K-2)\log\left(\frac{\zeta n^{-1}}{1+K\zeta n^{-1}}\right).
\end{aligned}$$

Therefore, one sees that all the parameters (ω_js and eigenvalues of $\boldsymbol{S}_j$s) remain in a bounded set. □

It is worth nothing that the aforementioned theorem is necessary, because neither condition (iv) nor condition (ii) hold for the entire space. From (28), it is easy to see that the Riemannian gradient of penalized log-likelihood with respect to one of the covariance matrices $\boldsymbol{S}_j$ for a single datapoint $\boldsymbol{y}_i$ is equal to

$$\nabla f_i(\boldsymbol{S}_j) = -\frac{w}{2}\boldsymbol{S}_j + \frac{w}{2}\boldsymbol{y}_i\boldsymbol{y}_i^T - \frac{\rho}{2n}\boldsymbol{S}_j + \frac{\beta}{2n}\boldsymbol{\Psi}. \tag{30}$$

Therefore if the norm of a component $||\boldsymbol{S}_j||$ go to infinity, then the norm of gradient go to infinity too. This means that the condition (iv) does not hold for the entire space. Let the norm of some components are bounded and the norm of one component goes to infinity, then it is clear that the condition (ii) can not hold either.
Since we proved the parameters remain in a compact subset of the underlying Riemannian manifold, we may invoke the following theorem:

Theorem 7 (Boumal et al. [10]) *Let $\mathcal{M}$ be a compact Riemannian submanifold of a Euclidean space. Let* Ret *be a retraction on M. If f has a Euclidean Lipschitz continuous gradient in the convex hull of $\mathcal{M}$, then the function satisfies the Lipschitz growth bound with some constant L for all retractions.*

We have shown above that the iterations of SGD for penalized log-likelihood stay within a compact set. It is also easy to see that the objective has a Euclidean Lipschitz continuous gradient on this set. Therefore, we can invoke Theorem 7 to show that the objective function satisfies condition (i) needed by Theorems 4 and 5. Furthermore, the objective function has a G-bounded gradient in this compact set and the iterations stay within it. Therefore, condition (iv) needed for Theorem 5 also holds. We summarize this result in the following corollary.

Corollary 3 *Assume SGD is used for optimizing the penalized log-likelihood of* GMM, *which is given by*

$$f\Big(\{\boldsymbol{S}_j \succ 0\}_{j=1}^K, \{\eta_j\}_{j=1}^{K-1}\Big) = \frac{1}{n}\sum_{i=1}^n f_i\Big(\{\boldsymbol{S}_j \succ 0\}_{j=1}^K, \{\eta_j\}_{j=1}^{K-1}\Big),$$

where f_i is as in (26). Then, the gradient of the objective after T iterations with constant step-size equal to $\eta_t = c/\sqrt{T}$ satisfies

$$\min_{1\le t\le T} \mathbb{E}\left[\|\nabla f^t\left(\left\{\boldsymbol{S}_j \succ 0\right\}_{j=1}^K, \{\eta_j\}_{j=1}^{K-1}\right)\|^2\right]$$
$$\le \frac{1}{\sqrt{T}}\left(\frac{f^* - f^0}{c} + \frac{Lc}{2}G^2\right) = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$

where f^t is the penalized objective evaluated at the value of parameters after t iterations; f^* is the value of penalized objective at its optimum; f^0 is the value of the objective at its initial point; L is the Lipschitz-growth bound constant; and G is the constant for the G-bounded condition of the gradient.

6 Experiments

In all experiments, the parameters of the penalizer in (13) are $\rho = \kappa = 0.01$ and $\alpha = \beta = 1$. The parameter $\boldsymbol{\Lambda}$ is set to 0.01 of sample covariance of the data and $\boldsymbol{\lambda}$ is sample mean of the data. The parameter ζ of the penalizer in (17) is set to 1. We initialize the mixture parameters using `k-means++` [3] by testing 30 different initial candidate and choosing the one with the best cost function. All methods stop when the difference between cost functions falls below 10^{-6}.

In order to show the efficacy of SGD, we fix the step-size rule in all experiments. We use exponential decay for the step-size. Given the maximum number of epochs, we set the starting step-size to 1 and the last step-size to 10^{-3}. The batch size is set to be equal to the dimensionality of data.

For the deterministic Riemannian optimization methods, we use exponential map and parallel transport as they lead to superior performance compared to other kinds of retractions and vector transports. For Riemannian SGD, we report the result of using Euclidean retraction. We also tested a more expensive exponential map and a different positivity-preserving retraction [21]. However, we observed no difference in cost function decrease as a function of gradient evaluations.

Two typical line-search methods are used in practice, one is Armijo rule and the other is line-search algorithm satisfying Wolfe conditions. For the case of LBFGS method, it is more common to use Wolfe line-search because it can guarantee that each step of LBFGS creates a descent direction [35]. In the following subsection, we give a short overview of manifold optimization algorithms used in this paper. Then in the next subsection, we explain the details of Wolfe line-search. We present the simulation results in the last subsection.

6.1 Manifold optimization algorithms

Algorithm 1 sketches a generic batch manifold optimization algorithm used in this paper. Two common optimization algorithms that fall into this generic one are Riemannian CG and Riemannian LBFGS. Key concepts needed for implementing the algorithms were explained in Sect. 2. Pseudocode for manifold SGD is shown in Algorithm 2.

Algorithm 3 is the pseudocode we used for implementing our Riemannian LBFGS algorithm. There are different variants of Riemannian LBFGS algorithm differ on where to apply the vector transport. We tested several variants and observed that the variant presented in Algorithm 3 which is the same as that of [38] achieved the best result. The vector transport used in our case of Algorithm 3 is the parallel transport. The adjoint of the parallel transport $\mathcal{T}_{x,y}^{*}$ is equal to the inverse of it $\mathcal{T}_{x,y}^{-1}$ which is also equal to $\mathcal{T}_{y,x}$. We use Riemannian CG implementation of the ManOpt toolbox [9].

Algorithm 1 Sketch of batch optimization algorithms (CG, LBFGS) on manifold

```
Given: Riemannian manifold $\mathcal{M}$ with Riemannian metric $g$; vector transport $\mathcal{T}$ on $\mathcal{M}$; retraction Ret; initial value $x_0$; a smooth function $f$
for $t = 0, 1, \dots$ do
    Obtain a descent direction $\xi_t$ based on stored information and $\nabla f(x_t)$ using defined $g$ and $\mathcal{T}$
    Use line-search to find $\alpha$ such that it satisfies appropriate conditions
    Calculate $x_{t+1} = \mathrm{Ret}_{x_t}(\alpha \xi_t)$
    Based on the memory and need of algorithm store $x_t$, $\nabla f(x_t)$ and $\alpha \xi_t$
end for
return $x_{t+1}$
```

Algorithm 2 Manifold SGD

```
Given: Smooth manifold $\mathcal{M}$ with retraction Ret; initial value $x_0$; a function $f$ that is the average of $n$ smooth functions $f(.) = \frac{1}{n}\sum_{i=1}^{n} f_i(.)$
for $t = 0, 1, \dots$ do
    The set $\mathcal{I}_t$ contains the indices of the data in the $t$th batch
    Obtain the direction $\xi_t = -\frac{1}{|\mathcal{I}_t|}\sum_{i \in \mathcal{I}_t} \nabla f_i(x_t)$
    Use a step-size rule to choose the step-size $\alpha_t$
    Calculate $x_{t+1} = \mathrm{Ret}_{x_t}(\alpha_t \xi_t)$
end for
return $x_{t+1}$
```

6.2 Wolfe line-search

The first Wolfe condition is a sufficient-decrease condition and is given by

$$f(\mathrm{Ret}_{x_t}(\alpha \xi_t)) \leq f(x_t) + c_1 \alpha Df(x_t)\xi_t,$$

where $0 < c_1 < 1$ is a constant typically chosen to be around 10^{-4} for LBFGS. This condition alone does not ensure that the algorithm makes sufficient progress. Another condition called curvature condition is needed,

$$Df(\mathrm{Ret}_{x_t}(\alpha \xi_t))\mathcal{T}_{x_t, \mathrm{Ret}_{x_t}(\alpha \xi_t)}(\xi_t) \geq c_2 Df(x_t)\xi_t, \tag{31}$$

where $c_2 > c_1$ is a constant smaller than 1 (around 0.9 for LBFGS). Practical line-search algorithms usually satisfy *strong Wolfe conditions*, where (31) is replaced by the stronger condition:

$$|Df(\mathrm{Ret}_{x_t}(\alpha \xi_t))\mathcal{T}_{x_t, \mathrm{Ret}_{x_t}(\alpha \xi_t)}(\xi_t)| \leq c_2 |Df(x_t)\xi_t|.$$

Algorithm 4 summarizes a line-search algorithm satisfying strong Wolfe conditions based on the Euclidean algorithm explained in [31]. The algorithm is divided into two phases: bracketing and zooming. In the bracketing phase, an interval is found that contains a point satisfying the strong Wolfe condition. Next, in the zooming phase, the actual point is found. Theory behind why this algorithm is guaranteed to find a step-length satisfying (strong) Wolfe conditions can be found in [31]. For the interpolation and extrapolation steps of the line-search one can find the minimum of a cubic polynomial approximation to the function in an interval. For cubic polynomial

Algorithm 3 Riemannian LBFGS

```
Given: Riemannian manifold $\mathcal{M}$ with Riemannian metric $g$; vector transport $\mathcal{T}$ on $\mathcal{M}$; retraction Ret;
initial value $x_0$; a smooth function $f$
Set initial $H_{\text{diag}} = 1/\sqrt{g_{x_0}(\nabla f(x_0), \nabla f(x_0))}$
for $t = 0, 1, \ldots$ do
    Obtain the descent direction $\xi_t \leftarrow \textsc{Desc}(-\nabla f(x_t), t)$
    Use line-search to find $\alpha$ such that it satisfies Wolfe conditions
    Calculate $x_{t+1} = \text{Ret}_{x_t}(\alpha\xi_t)$
    Define $s_{t+1} = \mathcal{T}_{x_t,x_{t+1}}(\alpha\xi_t)$
    Define $y_{t+1} = \nabla f(x_{t+1}) - \mathcal{T}_{x_t,x_{t+1}}(\nabla f(x_t))$
    Update $H_{\text{diag}} = g_{x_{t+1}}(s_{t+1}, y_{t+1})/g_{x_{t+1}}(y_{t+1}, y_{t+1})$
    Store $y_{t+1}$; $s_{t+1}$; $g_{x_{t+1}}(s_{t+1}, y_{t+1})$; $g_{x_{t+1}}(s_{t+1}, s_{t+1})$; $H_{\text{diag}}$
end for
return $x_{t+1}$
function $\textsc{Desc}(p, t)$ //obtaining the descent direction by unrolling the BFGS method
if $t > 0$ then
    $\tilde{p} = p - \frac{g_{x_t}(s_t, p)}{g_{x_t}(y_t, s_t)} y_t$
    $\hat{p} = \mathcal{T}_{x_{t-1},x_t}\textsc{Desc}(\mathcal{T}^*_{x_{t-1},x_t}\tilde{p}, t-1)$
                                // $\mathcal{T}^*_{x,y}$ is the adjoint of $\mathcal{T}_{x,y}$ [35] (defined by
                                // $g_y(v, T_{x,y}u) = g_x(u, T^*_{x,y}v)\ \forall u \in T_x\mathcal{M}, v \in T_y\mathcal{M}$)
    return $\hat{p} - \frac{g_{x_t}(y_t, \hat{p})}{g_{x_t}(y_t, s_t)} s_t + \frac{g_{x_t}(s_t, s_t)}{g_{x_t}(y_t, s_t)} p$
else
    return $H_{\text{diag}}\,p$
end if
end function
```

interpolation, we approximate the function by a cubic polynomial so that the function $\phi(\cdot)$ and its gradient $\phi'(\cdot)$ matches the function value and the gradient of the cubic polynomial at the end-points of the interval. For extrapolation, we use the function and gradient at 0 and at the end-point. To ensure numerical stability, the interval wherein the minimum of the cubic polynomial is computed in the interpolation phase is chosen to be smaller than the actual interval so to have certain distances from the end-points of the interval (we choose the distance to be 0.1 times the interval length). The interval for the extrapolation is assumed to be between 1.1 and 10 times the value of the current point.

The initial step-length α_1 can be guessed using the previous function and gradient information. We propose the following choice that is quite effective:

$$\alpha_1 = 2\frac{f(x_t) - f(x_{t-1})}{Df(x_t)\xi_t}. \tag{32}$$

Equation (32) is obtained by finding α^* that minimizes a quadratic approximation of the function along the geodesic through the previous point (based on $f(x_{t-1})$, $f(x_t)$ and $Df(x_{t-1})\xi_{t-1}$):

$$\alpha^* = 2\frac{f(x_t) - f(x_{t-1})}{Df(x_{t-1})\xi_{t-1}}. \tag{33}$$

Then, assuming that first-order change will be the same as in the previous step, we write

Algorithm 4 Wolfe line-search

```
1: Given: Current point $x_t$ and descent direction $\xi_t$
2: $\phi(\alpha) \leftarrow f(R_{x_t}(\alpha\xi_t))$; $\phi'(\alpha) \leftarrow \alpha Df(x_t)\xi_t$
3: $\alpha_0 \leftarrow 0, \alpha_1 > 0$ and $i \leftarrow 0$.
4: while $i \leq i_{\max}$ do
5:     $i \leftarrow i+1$
6:     if $\phi(\alpha_i) > \phi(0) + c_1\alpha_i\phi'(0)$ or $\phi(\alpha_i) \geq \phi(\alpha_{i-1}),\ i > 1$ then
7:         $\alpha_{\text{low}} = \alpha_{i-1}$ and $\alpha_{\text{hi}} = \alpha_i$
8:         break
9:     else if $|\phi'(\alpha_i)| \leq c_2|\phi'(0)|$ then return $\alpha_i$
10:     else if $\phi'(\alpha_i) \geq 0$ then
11:         $\alpha_{\text{low}} = \alpha_i$ and $\alpha_{\text{hi}} = \alpha_{i-1}$
12:         break
13:     else
14:         EXTRAPOLATE to find $\alpha_{i+1} > \alpha_i$
15:     end if
16: end while
17: Call ZOOMINGPHASE
```

Algorithm 5 ZOOMINGPHASE

```
1: while $i \leq i_{\max}$ do
2:     $i \leftarrow i+1$
3:     INTERPOLATE to find $\alpha_i \in (\alpha_{\text{low}}, \alpha_{\text{hi}})$
4:     if $\phi(\alpha_i) > \phi(0) + c_1\alpha_i\phi'(0)$ or $\phi(\alpha_i) \geq \phi(\alpha_{\text{low}})$ then
5:         $\alpha_{\text{hi}} \leftarrow \alpha_i$
6:     else
7:         if $|\phi'(\alpha_i)| \leq c_2|\phi'(0)|$ then return $\alpha_i$
8:         else if $\phi'(\alpha_i)(\alpha_{\text{hi}} - \alpha_{\text{low}}) \geq 0$ then
9:             $\alpha_{\text{hi}} \leftarrow \alpha_{\text{low}}$
10:         end if
11:         $\alpha_{\text{low}} \leftarrow \alpha_i$
12:     end if
13: end while
14: return failure
```

$$\alpha^* Df(x_{t-1})\xi_{t-1} \approx \alpha_1 Df(x_t)\xi_t. \tag{34}$$

Combining (33) and (34), we obtain our procedure of selection α_1 expressed in (32). Nocedal and Wright [31] suggest using either α^* of (33) as the initial step-length, or using (34) where α^* is set equal to the step-length obtained in the line-search at the previous point. We observed the our choice (32) proposed above leads to substantially better performance than these other two approaches.

6.3 Simulation results

In the first experiment, the effect of the problem reformulation of Sect. 3 is investigate. This effect is shown if Fig. 2. The left plot is the result of optimization for a single Gaussian and the right plot is the result for GMM with seven components. It can be seen that the reformulation has significant effect on the convergence speed.

Simulated data In the next set of experiments, we evaluate the statistical properties of different methods in recovering true parameters of underlying distribution. To this end, we sample from GMMs with known means and covariance matrices. It is know that the performance of EM algorithm depends on the degree of separation of the components in the mixture models [26,44]. We use the method of [12] to generate the parameters of GMMs with given degree of separation. In this method, the mean of components satisfy the following inequality:

$$\forall_{i \neq j} \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_j\| \geq c \max_{i,j} \left\{ \operatorname{tr}(\boldsymbol{\Sigma}_i), \operatorname{tr}(\boldsymbol{\Sigma}_j) \right\}$$

where c shows the degree of separation between different components. We use three different degrees of separation in our experiments: $c = 0.2$ (low separation), $c = 1$ (mid separation) and $c = 5$ high separation. Averaged negative penalized log-likelihood cost function of different methods for $d = 5$ and $K = 5$ are shown in Table 2. The eccentricity of the covariance matrices, i.e. the ration of the largest to the smallest eigenvalue, is set to $e = 10$. The numbers are reported for different amount of data $10d^2$, $100d^2$ and $1000d^2$, and different iterations 5, 20, 50, and also last iteration shown as *END* in the table. As it can be seen in the table, Riemannian SGD not only outperforms other methods in terms of convergence speed, but it also achieves better cost functions which means it can reach better local minima of the cost function.

To evaluate how well different methods recover the parameters of underlying distribution, we evaluate the estimation error of different methods. The estimated error for the mean vectors and the covariance matrices are shown in Tables 3 and 4, respectively. The estimation error for the mean vectors is calculated by summing norm of difference between estimated and true mean vectors, i.e.

$$\mathcal{E}_{\boldsymbol{\mu}} = \sum_{k=1}^{K} \|\boldsymbol{\mu}_k - \boldsymbol{\mu}_k^*\|,$$

where $\boldsymbol{\mu}_k$ is estimated and $\boldsymbol{\mu}_k^*$ is its corresponding true mean vectors. The estimation error for the covariance parameters is calculated by summing Frobenius norm of difference between estimated and true covariance matrices.

As it is clear from Tables 3 and 4, Riemannian SGD significantly outperforms EM algorithm and other optimization methods both in terms of speed and also accuracy. In compare to Table 2, where we reported the cost function, we here see more improvement. This behavior can be attributed to the observation that the SGD method has a tendency to converge to a point closer to the true parameters than that of other methods. An example of this behavior can be seen in Fig. 3, where all methods converge to pretty much the same cost function, but SGD converges to a better local minimum in the sense of closeness to the true parameters. Figure 4 shows an interesting behavior of SGD, where the method converges to a point with worst cost but closer to the true parameters.

In Table 5, we reported the estimation error for the case of smaller number of components $K = 2$. In this simple case, all methods perform well, while SGD outperforms other methods in one of the cases, that is low separation and small number of data.

Table 2 Averaged negative penalized log-likelihood cost function of GMMs for different methods, different amount of data and three different degrees of separation of components

c	Method	n	5	20	50	End
Low	EM	250	3.261	3.162	3.120	3.085
		2500	3.205	3.047	2.980	2.968
		25,000	3.524	3.390	3.323	3.301
	SGD	250	3.319	3.152	3.084	
		2500	2.996	2.965	2.970	
		25,000	3.312	3.303	3.301	
	LBFGS	250	3.316	3.130	3.097	3.097
		2500	3.247	3.051	2.985	2.979
		25,000	3.571	3.393	3.319	3.305
	CG	250	3.317	3.162	3.106	3.105
		2500	3.265	3.089	2.990	2.981
		25,000	3.584	3.413	3.326	3.305
Mid	EM	250	3.625	3.562	3.541	3.539
		2500	3.918	3.827	3.803	3.800
		25,000	3.739	3.644	3.643	3.599
	SGD	250	3.705	3.569	3.522	
		2500	3.809	3.794	3.782	
		25,000	3.565	3.548	3.548	
	LBFGS	250	3.692	3.538	3.526	3.525
		2500	3.986	3.815	3.799	3.794
		25,000	3.805	3.679	3.647	3.568
	CG	250	3.703	3.545	3.532	3.528
		2500	3.980	3.826	3.807	3.794
		25,000	3.801	3.687	3.658	3.599
High	EM	250	3.666	3.666	3.666	3.666
		2500	3.833	3.833	3.833	3.833
		25,000	3.520	3.520	3.520	3.520
	SGD	250	3.699	3.667	3.666	
		2500	3.834	3.833	3.833	
		25,000	3.521	3.520	3.520	
	LBFGS	250	3.677	3.666	3.666	3.666
		2500	3.848	3.833	3.833	3.833
		25,000	3.533	3.520	3.520	3.520
	CG	250	3.685	3.666	3.666	3.666
		2500	3.849	3.833	3.833	3.833
		25,000	3.536	3.520	3.520	3.520

The values for 5th, 20th, 50th and the last iteration are shown in the table. The results are averaged results for 10 different runs of algorithms for the data sampled from GMMs with $d = 5$, $K = 5$ and $e = 10$ (eccentricity of covariance matrices)

Table 3 Parameter estimation error of GMMs' mean vectors for different methods, different amount of data and three different degrees of separation of components

c	Method	n	5	20	50	End
Low	EM	250	3.779 ± 0.656	3.497 ± 0.759	3.328 ± 0.768	3.230 ± 0.807
		2500	3.096 ± 0.623	1.640 ± 0.767	0.705 ± 0.409	0.441 ± 0.208
		25,000	3.394 ± 0.731	1.964 ± 0.713	0.802 ± 0.637	0.175 ± 0.156
	SGD	250	3.886 ± 0.624	3.275 ± 0.557	2.941 ± 0.739	
		2500	1.043 ± 0.474	0.382 ± 0.096	0.468 ± 0.197	
		25,000	0.361 ± 0.192	0.198 ± 0.152	0.167 ± 0.143	
	LBFGS	250	3.986 ± 0.630	3.558 ± 0.787	3.395 ± 0.894	3.382 ± 0.885
		2500	3.628 ± 1.016	1.589 ± 0.817	1.012 ± 0.453	1.005 ± 0.599
		25,000	4.442 ± 1.880	2.366 ± 1.420	0.993 ± 1.038	0.405 ± 0.464
	CG	250	4.024 ± 0.624	3.574 ± 0.665	3.479 ± 0.854	3.469 ± 0.852
		2500	3.791 ± 1.105	2.218 ± 1.011	1.182 ± 0.468	1.023 ± 0.494
		25000	4.578 ± 1.787	2.825 ± 1.422	1.219 ± 1.140	0.443 ± 0.542
Mid	EM	250	2.808 ± 0.590	2.749 ± 0.711	2.800 ± 0.746	2.811 ± 0.758
		2500	1.983 ± 1.447	1.249 ± 1.266	1.017 ± 1.277	0.939 ± 1.146
		25000	2.601 ± 1.301	1.618 ± 1.723	1.526 ± 1.601	0.909 ± 1.323
	SGD	250	3.116 ± 0.732	2.571 ± 0.705	2.346 ± 0.785	
		2500	1.132 ± 1.151	0.675 ± 0.895	0.653 ± 0.894	
		25,000	0.559 ± 0.985	0.329 ± 0.768	0.328 ± 0.766	
	LBFGS	250	3.022 ± 0.758	2.692 ± 0.693	2.662 ± 0.688	2.663 ± 0.687
		2500	2.274 ± 1.502	1.230 ± 1.325	0.821 ± 0.942	0.631 ± 0.849
		25000	3.083 ± 1.209	2.050 ± 1.596	1.276 ± 1.289	0.702 ± 1.307
	CG	250	3.106 ± 0.696	2.689 ± 0.687	2.660 ± 0.668	2.664 ± 0.696
		2500	2.288 ± 1.546	1.338 ± 1.473	0.957 ± 1.013	0.634 ± 0.851
		25,000	3.074 ± 1.207	2.141 ± 1.535	1.484 ± 1.362	0.958 ± 1.413
High	EM	250	0.720 ± 0.150	0.720 ± 0.150	0.720 ± 0.150	0.720 ± 0.150
		2500	0.246 ± 0.057	0.246 ± 0.057	0.246 ± 0.057	0.246 ± 0.057
		25,000	0.078 ± 0.018	0.078 ± 0.018	0.078 ± 0.018	0.078 ± 0.018
	SGD	250	0.711 ± 0.153	0.719 ± 0.151	0.719 ± 0.149	
		2500	0.241 ± 0.054	0.245 ± 0.057	0.246 ± 0.056	
		25,000	0.089 ± 0.018	0.079 ± 0.019	0.079 ± 0.017	
	LBFGS	250	0.716 ± 0.142	0.720 ± 0.150	0.720 ± 0.150	0.720 ± 0.150
		2500	0.352 ± 0.161	0.246 ± 0.057	0.246 ± 0.057	0.246 ± 0.057
		25,000	0.169 ± 0.048	0.078 ± 0.018	0.078 ± 0.018	0.078 ± 0.018
	CG	250	0.737 ± 0.143	0.720 ± 0.150	0.720 ± 0.150	0.720 ± 0.150
		2500	0.374 ± 0.257	0.246 ± 0.057	0.246 ± 0.057	0.246 ± 0.057
		25,000	0.232 ± 0.168	0.078 ± 0.018	0.078 ± 0.018	0.078 ± 0.018

The values for 5th, 20th, 50th and the last iteration are shown in the table. The average and standard deviation are reported for 10 different runs of algorithms for the data sampled from GMMs with $d = 5$, $K = 5$ and $e = 10$ (eccentricity of covariance matrices)

Table 4 Parameter estimation error of GMMs' covariance matrices for different methods, different amount of data and three different degrees of separation of components

c	Method	n	5	20	50	End
Low	EM	250	2.585 ± 0.463	2.473 ± 0.540	2.462 ± 0.673	2.506 ± 0.676
		2500	2.236 ± 0.606	1.338 ± 0.615	0.818 ± 0.520	0.567 ± 0.370
		25,000	2.247 ± 0.416	1.625 ± 0.405	0.848 ± 0.510	0.274 ± 0.336
	SGD	250	2.646 ± 0.439	2.370 ± 0.581	2.336 ± 0.690	
		2500	1.001 ± 0.348	0.490 ± 0.247	0.642 ± 0.329	
		25,000	0.597 ± 0.359	0.279 ± 0.230	0.224 ± 0.190	
	LBFGS	250	2.639 ± 0.454	2.604 ± 0.660	2.619 ± 0.657	2.624 ± 0.661
		2500	2.453 ± 0.844	1.609 ± 1.061	1.046 ± 0.377	0.986 ± 0.411
		25,000	2.377 ± 0.580	1.605 ± 0.667	0.737 ± 0.720	0.375 ± 0.348
	CG	250	2.626 ± 0.431	2.568 ± 0.566	2.535 ± 0.705	2.531 ± 0.716
		2500	2.406 ± 0.726	1.791 ± 0.960	1.206 ± 0.581	1.058 ± 0.495
		25,000	2.424 ± 0.515	1.895 ± 0.778	0.894 ± 0.782	0.394 ± 0.391
Mid	EM	250	2.149 ± 0.665	2.133 ± 0.731	2.096 ± 0.722	2.083 ± 0.718
		2500	1.294 ± 0.760	0.900 ± 0.646	0.798 ± 0.629	0.745 ± 0.556
		25000	1.361 ± 0.462	0.725 ± 0.638	0.711 ± 0.626	0.467 ± 0.565
	SGD	250	2.416 ± 1.009	2.100 ± 0.675	1.956 ± 0.628	
		2500	0.901 ± 0.644	0.682 ± 0.564	0.653 ± 0.525	
		25,000	0.418 ± 0.647	0.232 ± 0.376	0.232 ± 0.373	
	LBFGS	250	2.341 ± 1.046	2.004 ± 0.726	2.015 ± 0.744	2.022 ± 0.738
		2500	1.468 ± 0.750	0.878 ± 0.619	0.704 ± 0.440	0.609 ± 0.381
		25,000	1.752 ± 0.340	1.112 ± 0.685	0.817 ± 0.639	0.353 ± 0.513
	CG	250	2.366 ± 1.077	2.004 ± 0.728	2.019 ± 0.731	2.008 ± 0.748
		2500	1.538 ± 0.845	0.912 ± 0.687	0.805 ± 0.533	0.611 ± 0.382
		25,000	1.763 ± 0.340	1.167 ± 0.653	0.851 ± 0.639	0.482 ± 0.587
High	EM	250	0.928 ± 0.303	0.928 ± 0.303	0.928 ± 0.303	0.928 ± 0.303
		2500	0.355 ± 0.086	0.355 ± 0.086	0.355 ± 0.086	0.355 ± 0.086
		25,000	0.095 ± 0.020	0.095 ± 0.020	0.095 ± 0.020	0.095 ± 0.020
	SGD	250	1.207 ± 0.389	0.945 ± 0.306	0.933 ± 0.303	
		2500	0.373 ± 0.088	0.358 ± 0.086	0.356 ± 0.087	
		25,000	0.113 ± 0.023	0.095 ± 0.019	0.095 ± 0.019	
	LBFGS	250	0.948 ± 0.294	0.928 ± 0.303	0.928 ± 0.303	0.928 ± 0.303
		2500	0.603 ± 0.335	0.356 ± 0.086	0.355 ± 0.086	0.355 ± 0.086
		25,000	0.298 ± 0.143	0.096 ± 0.020	0.096 ± 0.020	0.096 ± 0.020
	CG	250	1.049 ± 0.360	0.927 ± 0.303	0.927 ± 0.303	0.927 ± 0.303
		2500	0.628 ± 0.434	0.355 ± 0.087	0.355 ± 0.087	0.355 ± 0.087
		25,000	0.414 ± 0.362	0.095 ± 0.020	0.095 ± 0.020	0.095 ± 0.020

The values for 5th, 20th, 50th and the last iteration are shown in the table. The average and standard deviation are reported for 10 different runs of algorithms for the data sampled from GMMs with $d = 5$, $K = 5$ and $e = 10$

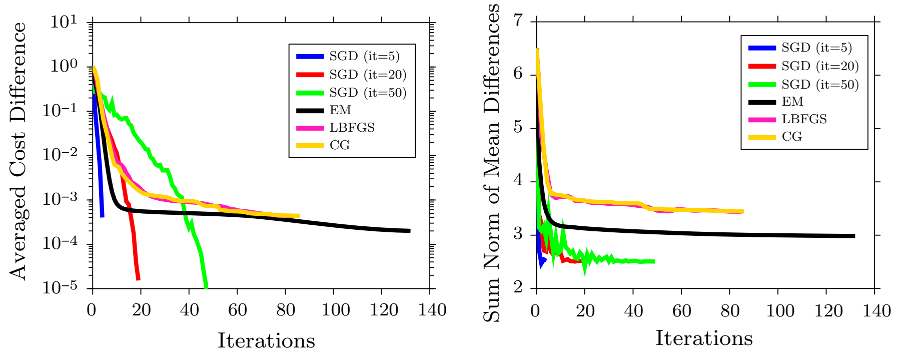

Fig. 3 Comparison of optimization methods on the simulated data ($d = 5$, $n = 25{,}000$, $e = 10$, mid separation). Left plot shows best cost minus current cost values as the function of number of function and gradient evaluations. Right plot shows sum norm of distances between estimated and true mean parameters of GMM components

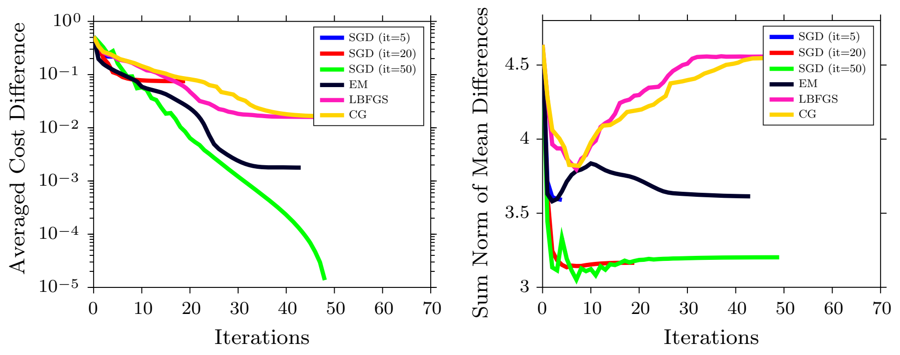

Fig. 4 Comparison of optimization methods on the simulated data ($d = 5$, $n = 250$, $e = 10$, mid separation). Left plot shows best cost minus current cost values as the function of number of function and gradient evaluations. Right plot shows sum norm of distances between estimated and true mean parameters of GMM components

Since components with low eccentricity have more overlap, we evaluate the performance where the components have no eccentricity, that is the case where covariance matrices are spherical. The results shown in Table 6 illustrate the fact that more overlap leads to slower convergence for optimization methods. The mid separation is the only place where we see other optimization methods outperform SGD. We attribute this behavior to the flatness of the objective function near optimum. In such cases, stochastic gradient descent would suffer from slow convergence behavior. An example of the result happening for the case of mid separation is shown in Fig. 5. The slow convergence of EM algorithm is clear from this figure. Riemannian LBFGS and CG algorithms estimate curvature information of the cost function and therefore they have faster convergence behavior. It is worth nothing that although SGD have relatively larger estimation error but its density is very close to the density of true distribution, because there exist some parameters with large distance from the true distribution that lead to almost the same density.

Table 5 Parameter estimation error of GMMs' mean vectors for different methods, different amount of data and three different degrees of separation of components

c	Method	n	5	20	50	End
Low	EM	250	0.486 ± 0.137	0.344 ± 0.222	0.306 ± 0.223	0.290 ± 0.222
		2500	0.783 ± 0.273	0.200 ± 0.154	0.072 ± 0.029	0.056 ± 0.015
		25,000	0.682 ± 0.235	0.233 ± 0.271	0.076 ± 0.146	0.016 ± 0.006
	SGD	250	0.400 ± 0.199	0.222 ± 0.150	0.182 ± 0.066	
		2500	0.062 ± 0.018	0.057 ± 0.015	0.056 ± 0.015	
		25,000	0.023 ± 0.008	0.016 ± 0.007	0.016 ± 0.006	
	LBFGS	250	0.486 ± 0.192	0.319 ± 0.218	0.287 ± 0.216	0.281 ± 0.211
		2500	1.055 ± 0.455	0.496 ± 0.516	0.146 ± 0.201	0.056 ± 0.015
		25,000	0.913 ± 0.520	0.426 ± 0.389	0.173 ± 0.281	0.016 ± 0.006
	CG	250	0.530 ± 0.163	0.363 ± 0.220	0.290 ± 0.219	0.282 ± 0.212
		2500	1.081 ± 0.460	0.525 ± 0.493	0.181 ± 0.258	0.056 ± 0.015
		25,000	0.965 ± 0.539	0.552 ± 0.493	0.195 ± 0.298	0.016 ± 0.006
Mid	EM	250	0.237 ± 0.158	0.143 ± 0.067	0.135 ± 0.065	0.135 ± 0.065
		2500	0.147 ± 0.252	0.117 ± 0.222	0.049 ± 0.019	0.045 ± 0.019
		25,000	0.208 ± 0.422	0.139 ± 0.397	0.012 ± 0.003	0.012 ± 0.003
	SGD	250	0.173 ± 0.080	0.145 ± 0.059	0.137 ± 0.063	
		2500	0.047 ± 0.019	0.045 ± 0.018	0.045 ± 0.018	
		25,000	0.019 ± 0.003	0.014 ± 0.004	0.013 ± 0.004	
	LBFGS	250	0.208 ± 0.145	0.136 ± 0.065	0.135 ± 0.065	0.135 ± 0.065
		2500	0.144 ± 0.261	0.047 ± 0.017	0.045 ± 0.018	0.045 ± 0.018
		25,000	0.233 ± 0.430	0.136 ± 0.387	0.013 ± 0.003	0.012 ± 0.003
	CG	250	0.210 ± 0.141	0.135 ± 0.064	0.135 ± 0.065	0.135 ± 0.065
		2500	0.155 ± 0.256	0.047 ± 0.017	0.045 ± 0.019	0.045 ± 0.019
		25000	0.247 ± 0.423	0.067 ± 0.170	0.012 ± 0.003	0.012 ± 0.003
High	EM	250	0.122 ± 0.037	0.122 ± 0.037	0.122 ± 0.037	0.122 ± 0.037
		2500	0.042 ± 0.010	0.042 ± 0.010	0.042 ± 0.010	0.042 ± 0.010
		25,000	0.012 ± 0.004	0.012 ± 0.004	0.012 ± 0.004	0.012 ± 0.004
	SGD	250	0.127 ± 0.040	0.122 ± 0.037	0.122 ± 0.037	
		2500	0.041 ± 0.011	0.042 ± 0.010	0.042 ± 0.010	
		25,000	0.018 ± 0.004	0.013 ± 0.005	0.013 ± 0.005	
	LBFGS	250	0.121 ± 0.036	0.122 ± 0.037	0.122 ± 0.037	0.122 ± 0.037
		2500	0.059 ± 0.028	0.042 ± 0.010	0.042 ± 0.010	0.042 ± 0.010
		25,000	0.021 ± 0.008	0.012 ± 0.004	0.012 ± 0.004	0.012 ± 0.004
	CG	250	0.127 ± 0.040	0.122 ± 0.037	0.122 ± 0.037	0.122 ± 0.037
		2500	0.072 ± 0.050	0.042 ± 0.010	0.042 ± 0.010	0.042 ± 0.010
		25,000	0.019 ± 0.020	0.012 ± 0.004	0.012 ± 0.004	0.012 ± 0.004

The values for 5th, 20th, 50th and the last iteration are shown in the table. The average and standard deviation are reported for 10 different runs of algorithms for the data sampled from GMMs with $d = 5$, $K = 2$ and $e = 10$

Table 6 Parameter estimation error of GMMs' mean vectors for different methods, different amount of data and three different degrees of separation of components

c	Method	n	5	20	50	End
Low	EM	250	0.611 ± 0.051	0.601 ± 0.057	0.587 ± 0.058	0.577 ± 0.088
		2500	0.540 ± 0.051	0.490 ± 0.048	0.475 ± 0.046	0.456 ± 0.066
		25,000	0.543 ± 0.039	0.472 ± 0.043	0.434 ± 0.048	0.381 ± 0.055
	SGD	250	0.630 ± 0.076	0.526 ± 0.028	0.481 ± 0.060	
		2500	0.341 ± 0.085	0.227 ± 0.064	0.206 ± 0.044	
		25000	0.164 ± 0.009	0.155 ± 0.012	0.163 ± 0.014	
	LBFGS	250	0.636 ± 0.069	0.601 ± 0.066	0.600 ± 0.073	0.602 ± 0.076
		2500	0.624 ± 0.096	0.555 ± 0.075	0.535 ± 0.070	0.541 ± 0.102
		25,000	0.720 ± 0.137	0.630 ± 0.139	0.570 ± 0.122	0.555 ± 0.111
	CG	250	0.645 ± 0.076	0.593 ± 0.059	0.604 ± 0.072	0.600 ± 0.077
		2500	0.643 ± 0.097	0.554 ± 0.072	0.533 ± 0.062	0.521 ± 0.084
		25,000	0.726 ± 0.141	0.634 ± 0.137	0.579 ± 0.124	0.550 ± 0.141
Mid	EM	250	0.781 ± 0.107	0.741 ± 0.109	0.741 ± 0.105	0.735 ± 0.120
		2500	0.604 ± 0.156	0.504 ± 0.157	0.439 ± 0.152	0.400 ± 0.176
		25,000	0.616 ± 0.094	0.518 ± 0.116	0.427 ± 0.125	0.187 ± 0.160
	SGD	250	0.848 ± 0.104	0.736 ± 0.119	0.694 ± 0.115	
		2500	0.549 ± 0.120	0.645 ± 0.050	0.606 ± 0.067	
		25,000	0.563 ± 0.070	0.547 ± 0.120	0.499 ± 0.145	
	LBFGS	250	0.852 ± 0.110	0.772 ± 0.117	0.757 ± 0.109	0.761 ± 0.112
		2500	0.824 ± 0.221	0.579 ± 0.201	0.496 ± 0.209	0.478 ± 0.215
		25,000	0.713 ± 0.110	0.536 ± 0.109	0.309 ± 0.176	0.133 ± 0.126
	CG	250	0.873 ± 0.109	0.784 ± 0.114	0.756 ± 0.118	0.750 ± 0.113
		2500	0.838 ± 0.230	0.607 ± 0.210	0.497 ± 0.196	0.446 ± 0.210
		25,000	0.730 ± 0.098	0.554 ± 0.109	0.434 ± 0.144	0.142 ± 0.122
High	EM	250	0.163 ± 0.024	0.163 ± 0.024	0.163 ± 0.024	0.163 ± 0.024
		2500	0.055 ± 0.006	0.055 ± 0.006	0.055 ± 0.006	0.055 ± 0.006
		25,000	0.017 ± 0.004	0.017 ± 0.004	0.017 ± 0.004	0.017 ± 0.004
	SGD	250	0.186 ± 0.037	0.163 ± 0.024	0.163 ± 0.024	
		2500	0.055 ± 0.007	0.055 ± 0.006	0.055 ± 0.006	
		25,000	0.019 ± 0.004	0.017 ± 0.004	0.017 ± 0.004	
	LBFGS	250	0.176 ± 0.042	0.163 ± 0.024	0.163 ± 0.024	0.163 ± 0.024
		2500	0.078 ± 0.034	0.055 ± 0.006	0.055 ± 0.006	0.055 ± 0.006
		25,000	0.041 ± 0.017	0.017 ± 0.004	0.017 ± 0.004	0.017 ± 0.004
	CG	250	0.190 ± 0.049	0.163 ± 0.024	0.163 ± 0.024	0.163 ± 0.024
		2500	0.085 ± 0.036	0.055 ± 0.006	0.055 ± 0.006	0.055 ± 0.006
		25,000	0.055 ± 0.044	0.017 ± 0.004	0.017 ± 0.004	0.017 ± 0.004

The values for 5th, 20th, 50th and the last iteration are shown in the table. The average and standard deviation are reported for 10 different runs of algorithms for the data sampled from GMMs with $d = 5$, $K = 5$ and $e = 1$ (spherical covariance matrices)

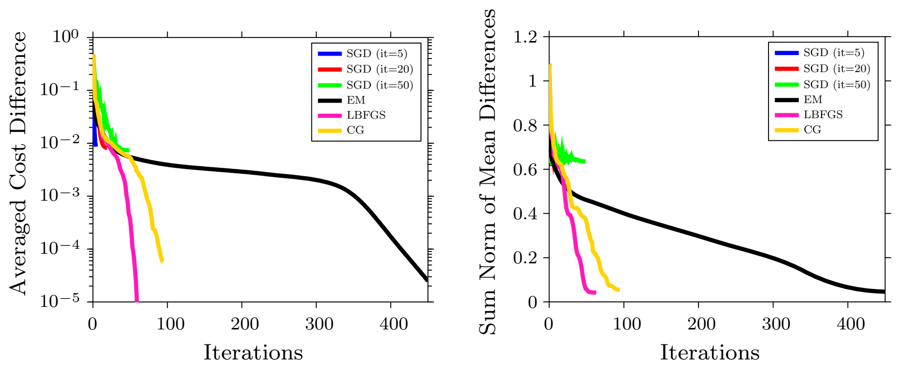

Fig. 5 Comparison of optimization methods on the simulated data ($d = 5$, $n = 25{,}000$, $e = 1$ and mid separation). Left plot shows best cost minus current cost values as the function of number of function and gradient evaluations. Right plot shows sum norm of distances between estimated and true mean parameters of GMM components

Table 7 Parameter estimation error of GMMs' mean vectors for different methods, different amount of data and three different degreess of separation of components

c	Method	n	5	20	50	End
Low	EM	4000	2.072 ± 0.476	1.112 ± 0.798	1.042 ± 0.726	1.015 ± 0.684
		40,000	1.490 ± 0.966	0.730 ± 0.989	0.654 ± 0.857	0.612 ± 0.814
		400,000	1.708 ± 0.536	0.943 ± 0.657	0.807 ± 0.665	0.408 ± 0.579
	SGD	4000	1.115 ± 0.799	0.804 ± 0.612	0.859 ± 0.551	
		40,000	0.465 ± 0.397	0.319 ± 0.358	0.232 ± 0.245	
		400,000	0.397 ± 0.393	0.146 ± 0.281	0.139 ± 0.269	
	LBFGS	4000	3.849 ± 1.129	1.472 ± 0.782	1.152 ± 0.772	1.146 ± 0.759
		40,000	3.869 ± 1.010	1.268 ± 0.971	0.604 ± 0.763	0.410 ± 0.556
		400,000	4.865 ± 1.595	1.423 ± 1.010	0.687 ± 0.771	0.137 ± 0.284
	CG	4000	4.629 ± 1.407	1.870 ± 0.766	1.029 ± 0.814	1.004 ± 0.741
		40,000	5.759 ± 2.138	1.531 ± 1.160	0.705 ± 0.899	0.432 ± 0.602
		400,000	6.001 ± 1.510	2.227 ± 1.227	0.673 ± 0.707	0.136 ± 0.282
Mid	EM	4000	0.938 ± 1.475	0.929 ± 1.445	0.920 ± 1.418	0.921 ± 1.422
		40,000	0.582 ± 1.367	0.578 ± 1.356	0.568 ± 1.325	0.553 ± 1.277
		400,000	0.479 ± 1.369	0.472 ± 1.348	0.467 ± 1.333	0.464 ± 1.321
	SGD	4000	0.941 ± 1.437	0.923 ± 1.427	0.923 ± 1.426	
		40,000	0.551 ± 1.262	0.552 ± 1.276	0.553 ± 1.278	
		400,000	0.091 ± 0.008	0.058 ± 0.006	0.053 ± 0.004	
	LBFGS	4000	1.890 ± 1.423	0.921 ± 1.414	0.916 ± 1.405	0.913 ± 1.394
		40,000	1.587 ± 1.620	0.883 ± 1.559	0.565 ± 1.315	0.556 ± 1.287
		400,000	0.991 ± 1.136	0.409 ± 1.147	0.107 ± 0.192	0.046 ± 0.004
	CG	4000	1.845 ± 1.408	0.920 ± 1.417	0.915 ± 1.402	0.914 ± 1.400

Table 7 continued

c	Method	n	5	20	50	End
		40,000	1.646 ± 1.461	0.852 ± 1.504	0.559 ± 1.296	0.541 ± 1.240
		400,000	1.276 ± 1.150	0.390 ± 1.085	0.046 ± 0.004	0.046 ± 0.004
High	EM	4000	0.488 ± 0.071	0.488 ± 0.071	0.488 ± 0.071	0.488 ± 0.071
		40,000	0.162 ± 0.027	0.162 ± 0.027	0.162 ± 0.027	0.162 ± 0.027
		400,000	0.046 ± 0.007	0.046 ± 0.007	0.046 ± 0.007	0.046 ± 0.007
	SGD	4000	0.502 ± 0.078	0.488 ± 0.072	0.488 ± 0.071	
		40,000	0.165 ± 0.026	0.162 ± 0.027	0.162 ± 0.027	
		400,000	0.088 ± 0.008	0.057 ± 0.008	0.052 ± 0.008	
	LBFGS	4000	1.435 ± 0.862	0.488 ± 0.071	0.488 ± 0.071	0.488 ± 0.071
		40,000	1.126 ± 0.665	0.162 ± 0.027	0.162 ± 0.027	0.162 ± 0.027
		400,000	0.900 ± 0.428	0.046 ± 0.007	0.046 ± 0.007	0.046 ± 0.007
	CG	4000	1.125 ± 0.656	0.488 ± 0.071	0.488 ± 0.071	0.488 ± 0.071
		40,000	1.284 ± 0.948	0.162 ± 0.027	0.162 ± 0.027	0.162 ± 0.027
		400,000	0.642 ± 0.377	0.046 ± 0.007	0.046 ± 0.007	0.046 ± 0.007

The values for 5th, 20th, 50th and the last iteration are shown in the table. The average and standard deviation are reported for 10 different runs of algorithms for the data sampled from GMMs with $d = 5$, $K = 5$ and $e = 20$

We also evaluate the performance of different algorithms for synthetic data of a larger dimension which is shown in Table 7. It can be seen that SGD significantly outperforms other methods in low separation case. Unlike previous results, we see in this table that all methods work similarly well in mid separation and high separation. This is because in higher dimensions, the components with high eccentricity are less likely to overlap.

Real data In the last set of experiments, we compare the performance of manifold optimization methods on the reformulated problem and EM on some real datasets. One of the datasets is a dataset of natural images [20]. The other three datasets called 'corel', 'yearpredict' and 'wine' data are taken from UCI machine learning dataset repository.[2] The results are shown in Figs. 6, 7, 8 and 9. The dimensionality d of data and number of data-points n are given in the figure legends.

It can be seen than deterministic manifold optimization methods achieve and outperforms the EM algorithm. The manifold SGD shows remarkable performance. This method leads to fast increase of the objective function in early iterations.

7 Conclusions and future work

In this paper, we proposed a reformulation for the GMM problem that can make Riemannian manifold optimization a powerful alternative to the EM algorithm for fitting

[2] Available via https://archive.ics.uci.edu/ml/datasets.

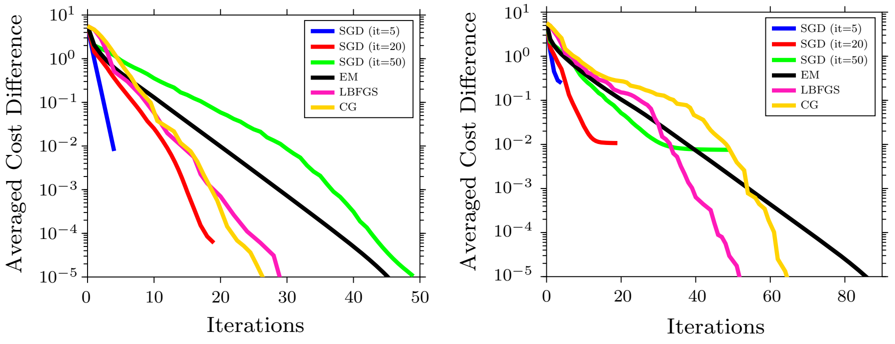

Fig. 6 Comparison of optimization methods on natural image data ($d = 35$, $n = 200{,}000$). Y-axis: best cost minus current cost values. X-axis: number of function and gradient evaluations. Right: 3 number of components. Left: 7 number of components

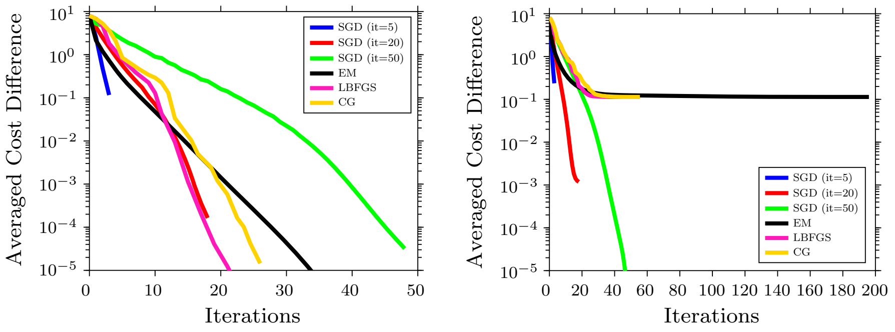

Fig. 7 Comparison of optimization methods on year predict data ($d = 90$, $n = 515{,}345$). Y-axis: best cost minus current cost values. X-axis: number of function and gradient evaluations. Right: 3 number of components. Left: 7 number of components

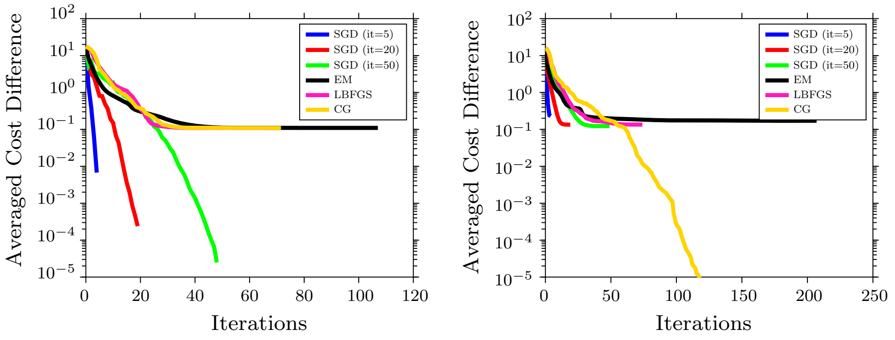

Fig. 8 Comparison of optimization methods on corel data ($d = 57$, $n = 68{,}040$). Y-axis: current objective values minus best objective. X-axis: number of function and gradient evaluations. Right: 3 number of components. Left: 7 number of components

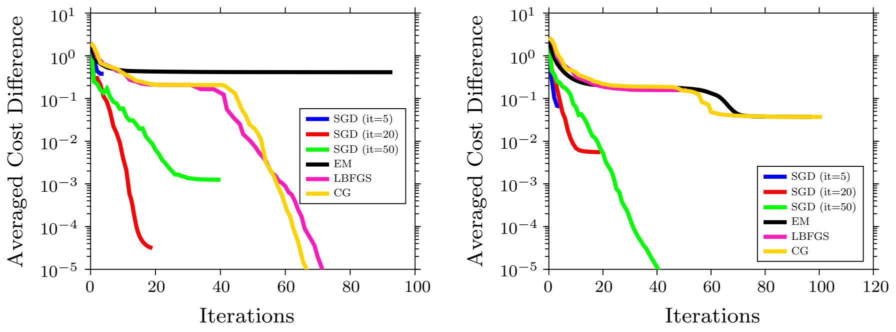

Fig. 9 Comparison of optimization methods on wine data ($d = 11$, $n = 6497$). Y-axis: current objective values minus best objective. X-axis: number of function and gradient evaluations. Right: 3 number of components. Left: 7 number of components

Gaussian mixture models. The deterministic manifold optimization methods can either match or outperform EM algorithm. Furthermore, we developed a global convergence theory for SGD on manifolds. We applied this theory to the GMM modeling. Experimentally Riemannian SGD for GMM shows remarkable convergence behavior, making it a potential candidate for large scale mixture modeling.

There are several venues for future works:

- Extension of Riemannian optimization to estimation in hidden Markov models.
- An exploration of manifold optimization for non-Gaussian mixture models.
- Study of richer priors for GMMs beyond the usual conjugate priors.
- Developing the Riemannian analog of the work of Balakrishnan et al. [4]. That is to show Riemannian optimization methods (especially Riemannian SGD) achieve optimal statistical properties for the case of GMMs.
- Developing almost sure (or high-probability) convergence results to the stationary points of the objective function.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton (2009)
2. Alvarez, F., Bolte, J., Brahic, O.: Hessian Riemannian gradient flows in convex programming. SIAM J. Control Optim. **43**(2), 477–501 (2004)
3. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1027–1035 (2007)
4. Balakrishnan, S., Wainwright, M.J., Yu, B.: Statistical guarantees for the EM algorithm: from population to sample-based analysis (2014). arXiv:1408.2156
5. Bhatia, R.: Positive Definite Matrices. Princeton University Press, Princeton (2007)
6. Bhojanapalli, S., Kyrillidis, A., Sanghavi, S.: Dropping convexity for faster semi-definite optimization. In: 29th Annual Conference on Learning Theory (COLT), pp. 530–582 (2016)
7. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2007)
8. Bonnabel, S.: Stochastic gradient descent on Riemannian manifolds. IEEE Trans. Autom. Control **58**(9), 2217–2229 (2013)
9. Boumal, N., Mishra, B., Absil, P.A., Sepulchre, R.: Manopt, a matlab toolbox for optimization on manifolds. J. Mach. Learn. Res. **15**(1), 1455–1459 (2014)

10. Boumal, N., Absil, P.A., Cartis, C.: Global rates of convergence for nonconvex optimization on manifolds (2016). arXiv:1605.08101v1
11. Burer, S., Monteiro, R.D., Zhang, Y.: Solving semidefinite programs via nonlinear programming. part I: transformations and derivatives. Tech. Rep. TR99-17, Department of Computational and Applied Mathematics, Rice University, Houston TX (1999)
12. Dasgupta, S.: Learning mixtures of Gaussians. In: 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 634–644 (1999)
13. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B **39**, 1–38 (1977)
14. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2000)
15. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning. Springer, Berlin (2001)
16. Ge, R., Huang, Q., Kakade, S.M.: Learning mixtures of Gaussians in high dimensions (2015). arXiv:1503.00424
17. Ghadimi, S., Lan, G.: Stochastic first- and zeroth-order methods for nonconvex stochastic programming. SIAM J. Optim. **23**(4), 2341–2368 (2013)
18. Hiai, F., Petz, D.: Riemannian metrics on positive definite matrices related to means. Linear Algebra Appl. **430**(11–12), 3105–3130 (2009)
19. Hiai, F., Petz, D.: Riemannian metrics on positive definite matrices related to means. ii. Linear Algebra Appl. **436**(7), 2117–2136 (2012)
20. Hosseini, R., Sra, S.: Matrix manifold optimization for Gaussian mixtures. In: Advances in Neural Information Processing Systems (NIPS), vol. 28, pp. 910–918 (2015)
21. Jeuris, B., Vandebril, R., Vandereycken, B.: A survey and comparison of contemporary algorithms for computing the matrix geometric mean. Electron. Trans. Numer. Anal. **39**, 379–402 (2012)
22. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Comput. **6**(2), 181–214 (1994)
23. Journée, M., Bach, F., Absil, P.A., Sepulchre, R.: Low-rank optimization on the cone of positive semidefinite matrices. SIAM J. Optim. **20**(5), 2327–2351 (2010)
24. Keener, R.W.: Theoretical Statistics. Springer Texts in Statistics. Springer, Berlin (2010)
25. Lee, J.M.: Introduction to Smooth Manifolds. Springer, Berlin (2012)
26. Ma, J., Xu, L., Jordan, M.I.: Asymptotic convergence rate of the EM algorithm for Gaussian mixtures. Neural Comput. **12**(12), 2881–2907 (2000)
27. McLachlan, G.J., Peel, D.: Finite Mixture Models. Wiley, New Yrok (2000)
28. Moitra, A., Valiant, G.: Settling the polynomial learnability of mixtures of Gaussians. In: 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 93–102 (2010)
29. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge (2012)
30. Naim, I., Gildea, D.: Convergence of the EM algorithm for Gaussian mixtures with unbalanced mixing coefficients. In: 29th International Conference on Machine Learning (ICML), pp. 1655–1662 (2012)
31. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, Berlin (2006)
32. Redner, R.A., Walker, H.F.: Mixture densities, maximum likelihood, and the EM algorithm. SIAM Rev. **26**, 195–239 (1984)
33. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker verification using adapted Gaussian mixture models. Digit. Signal Process. **10**(1–3), 19–41 (2000)
34. Ridolfi, A., Idier, J., Mohammad-Djafari, A.: Penalized maximum likelihood estimation for univariate normal mixture distributions. In: Actes du 17^e Colloque GRETSI, pp. 259–262 (1999)
35. Ring, W., Wirth, B.: Optimization methods on Riemannian manifolds and their application to shape space. SIAM J. Optim. **22**(2), 596–627 (2012)
36. Salakhutdinov, R., Roweis, S.T., Ghahramani, Z.: Optimization with EM and expectation-conjugate-gradient. In: 20th International Conference on Machine Learning (ICML), pp. 672–679 (2003)
37. Sra, S., Hosseini, R.: Geometric optimisation on positive definite matrices for elliptically contoured distributions. In: Advances in Neural Information Processing Systems (NIPS), vol. 26, pp. 2562–2570 (2013)
38. Sra, S., Hosseini, R.: Conic geometric optimization on the manifold of positive definite matrices. SIAM J. Optim. **25**(1), 713–739 (2015)
39. Udrişte, C.: Convex Functions and Optimization Methods on Riemannian Manifolds. Kluwer Academic, Dordrecht (1994)

40. Vanderbei, R.J., Benson, H.Y.: On formulating semidefinite programming problems as smooth convex nonlinear optimization problems. Tech. Rep. ORFE-99-01, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ (2000)
41. Vandereycken, B.: Low-rank matrix completion by Riemannian optimization. SIAM J. Optim. **23**(2), 1214–1236 (2013)
42. Wiesel, A.: Geodesic convexity and covariance estimation. IEEE Trans. Signal Process. **60**(12), 6182–89 (2012)
43. Wisdom, S., Powers, T., Hershey, J., Le Roux, J., Atlas, L.: Full-capacity unitary recurrent neural networks. In: Advances in Neural Information Processing Systems (NIPS), vol. 29, pp. 4880–4888 (2016)
44. Xu, L., Jordan, M.I.: On convergence properties of the EM algorithm for Gaussian mixtures. Neural Comput. **8**, 129–151 (1996)
45. Zhang, H., Sra, S.: First-order methods for geodesically convex optimization. In: 29th Annual Conference on Learning Theory (COLT), pp 1617–1638 (2016)
46. Zhang, H., Reddi, S., Sra, S.: Riemannian SVRG: Fast stochastic optimization on Riemannian manifolds. In: Advances in Neural Information Processing Systems (NIPS), vol. 29, pp. 4592–4600 (2016)