Proceedings of Machine Learning Research 1–2, 2021 ACML 2021

Vector Transport Free Riemannian LBFGS for Optimization on Symmetric Positive Definite Matrix Manifolds

Authors Anonymous EMAILS
Affiliations

Editors: Vineeth N Balasubramanian and Ivor Tsang

Abstract

This work concentrates on optimization on Riemannian manifolds. The Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) algorithm is a commonly used quasi-Newton method for numerical optimization in Euclidean spaces. Riemannian LBFGS (RLBFGS) is an extension of this method to Riemannian manifolds. RLBFGS involves computationally expensive vector transports as well as unfolding recursions using adjoint vector transports. In this article, we propose two mappings in the tangent space using the inverse second root and Cholesky decomposition. These mappings make both vector transport and adjoint vector transport identity and therefore isometric. Identity vector transport makes RLBFGS less computationally expensive and its isometry is also very useful in convergence analysis of RLBFGS. Moreover, under the proposed mappings, the Riemannian metric reduces to Euclidean inner product, which is much less computationally expensive. We focus on the Symmetric Positive Definite (SPD) manifolds which are beneficial in various fields such as data science and statistics. This work opens a research opportunity for extension of the proposed mappings to other well-known manifolds.

Keywords: LBFGS, Riemannian optimization, positive definite manifolds, isometric vector transport, quasi-Newton's method

1. Introduction

Various numerical optimization methods have appeared, for the Euclidean spaces, which can be categorized into first-order and second-order methods (Nocedal and Wright, 2006). Examples for the former category are steepest descent and gradient descent and for the latter group are Newton's method. Computation of the Hessian matrix is usually expensive in Newton's method encouraging many practical problems to either use quasi-Newton's methods for approximating the Hessian matrix or use non-linear conjugate gradient (Hestenes and Stiefel, 1952). The most well-known algorithm for quasi-Newton optimization is Broyden–Fletcher–Goldfarb–Shanno (BFGS) (Fletcher, 2013). Limited-memory BFGS (LBFGS) is a simplified version of BFGS which utilizes less memory (Nocedal, 1980; Liu and Nocedal, 1989). It has recursive unfoldings which approximate the descent directions in optimization.

Unlike Euclidean spaces in which the optimization direction lie in a linear coordinate system, Riemannian spaces have curvature in coordinates. Recently, extension of optimization methods from Euclidean spaces to Riemannian manifolds has been extensively noticed in the literature (Absil et al., 2009; Boumal, 2020). For example, Euclidean BFGS has been extended to Riemannian manifolds, named Riemannian BFGS (RBFGS), (Qi et al., 2010), its convergence has been proven (Ring and Wirth, 2012; Huang et al., 2015), and its properties have been analyzed in the literature (Seibert et al., 2013). As vector transport is computationally expensive in RBFGS, cautious RBFGS was proposed (Huang et al., 2016) which ignores the curvature condition in the Wolfe conditions (Wolfe, 1969) and only checks the Armijo condition (Armijo, 1966). Since the curvature condition guarantees that the approximation of Hessian remains positive definite, it compensates by checking a cautious condition (Li and Fukushima, 2001) before updating the approximation of Hessian. This cautious RBFGS has been used in the Manopt optimization toolbox (Boumal et al., 2014). Another approach is an extension of the Euclidean LBFGS to Riemannian manifolds, named Riemannian LBFGS (RLBFGS), using both Wolfe conditions (Wolfe, 1969) in linesearch can be found in (Sra and Hosseini, 2015, 2016; Hosseini and Sra, 2020). Some other direct extensions of Euclidean BFGS to Riemannian spaces exist (e.g., see (Ji, 2007, Chapter 7)).

In this paper, we address the computationally expensive parts of RLBFGS algorithm, which are computation of vector transports, their adjoints, and Riemannian metrics. To achieve this, we propose two mappings, in the tangent space, which make RLBFGS free of vector transport. One mapping uses inverse second root and the other uses Cholesky decomposition which is very efficient (Golub and Van Loan, 2013). The proposed mappings make both vector transport and adjoint vector transport, which are used in RLBFGS, identity. This reduction of transports to identity makes optimization much less expensive computationally. Moreover, as the vector transports become identity, they are isometric which is a suitable property mostly used in the convergence proofs of RBFGS and RLBFGS algorithms (Ring and Wirth, 2012; Huang et al., 2015). Furthermore, under the proposed mappings, the Riemannian metric reduces to Euclidean inner product which is much less computationally expensive. In this paper, we concentrate on the Symmetric Positive Definite (SPD) manifolds (Sra and Hosseini, 2015, 2016; Bhatia, 2009) which are very useful in data science and machine learning, such as in mixture models (Hosseini and Sra, 2020; Hosseini and Mash'al, 2015). This paper opens a new research path for extension of the proposed mappings to other well-known manifolds such as Grassmann and Stiefel (Edelman et al., 1998).

The remainder of this paper is organized as follows. Section 2 reviews the notations and technical background on Euclidean LBFGS, Wolfe conditions, Riemannian LBFGS, and SPD manifold. The proposed mappings using inverse second root and Cholesky decomposition are introduced in Sections 3 and 4, respectively. Simulation results are reported in Section 5. Finally, Section 6 concludes the paper and proposes the possible future directions.

2. Background and Notations

2.1. Euclidean BFGS and LBFGS

Consider minimization of the cost function $f(\boldsymbol{\Sigma})$ where the point $\boldsymbol{\Sigma}$ belongs to some domain. In Newton's method, the descent direction $\boldsymbol{p}_k$ at the iteration k is calculated as (Nocedal and Wright, 2006):

$$\boldsymbol{B}_k \boldsymbol{p}_k = -\nabla f(\boldsymbol{\Sigma}_k) \implies \boldsymbol{p}_k = -\boldsymbol{B}_k^{-1} \nabla f(\boldsymbol{\Sigma}_k), \tag{1}$$

where $\boldsymbol{B}_k$ is the Hessian or approximation of Hessian and $\nabla f(\boldsymbol{\Sigma}_k)$ is the gradient of function at iteration k. The Euclidean BFGS method is a quasi-Newton's method which approximates the Hessian matrix as (Fletcher, 2013; Nocedal and Wright, 2006):

$$\boldsymbol{B}_{k+1} := \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^\top}{\boldsymbol{y}_k^\top \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \boldsymbol{s}_k^\top \boldsymbol{B}_k^\top}{\boldsymbol{s}_k^\top \boldsymbol{B}_k \boldsymbol{s}_k}, \tag{2}$$

where $\boldsymbol{s}_k := \boldsymbol{\Sigma}_{k+1} - \boldsymbol{\Sigma}_k$ and $\boldsymbol{y}_k := \nabla f(\boldsymbol{\Sigma}_{k+1}) - \nabla f(\boldsymbol{\Sigma}_k)$. The descent direction is then calculated using Eq. (1).

The Euclidean LBFGS calculates the descent direction recursively where it uses the approximation of the inverse Hessian as follows (Nocedal, 1980; Liu and Nocedal, 1989):

$$\boldsymbol{H}_{k+1} := \boldsymbol{V}_k^\top \boldsymbol{H}_k \boldsymbol{V}_k + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^\top, \tag{3}$$

where $\boldsymbol{H}_k$ denotes the approximation of the inverse of the Hessian at iteration k, $\rho_k := 1/(\boldsymbol{y}_k^\top \boldsymbol{s}_k)$, and $\boldsymbol{V}_k := \boldsymbol{I} - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^\top$ in which $\boldsymbol{I}$ denotes the identity matrix. The LBFGS algorithm updates the approximation of the inverse of the Hessian matrix recursively and for that it always stores a memory window of pairs $\{\boldsymbol{y}_k, \boldsymbol{s}_k\}$ (Liu and Nocedal, 1989).

2.2. Linesearch and Wolfe Conditions

After finding the descent direction $\boldsymbol{p}_k$ at each iteration k of optimization, one needs to know what step size α_k should be taken in that direction. Linesearch should be performed to find the largest step, for faster progress, satisfying Wolfe conditions (Wolfe, 1969):

$$f(\boldsymbol{\Sigma}_k + \alpha_k) \leq f(\boldsymbol{\Sigma}_k) + c_1 \alpha_k \boldsymbol{p}_k^\top \nabla f(\boldsymbol{\Sigma}_k), \tag{4}$$

$$-\boldsymbol{p}_k^\top \nabla f(\boldsymbol{\Sigma}_k + \alpha_k \boldsymbol{p}_k) \leq -c_2 \boldsymbol{p}_k^\top \nabla f(\boldsymbol{\Sigma}_k), \tag{5}$$

where $0 < c_1 < c_2 < 1$ and are recommended to be $c_1 = 10^{-1}$ and $c_2 = 0.9$ (Nocedal and Wright, 2006). The former condition is the Armijo condition to check if cost decreases sufficiently (Armijo, 1966) while the latter is the curvature condition making sure that the slope is reduced sufficiently in a way that the approximation of Hessian remains positive definite. Note that there also exists a strong curvature condition, i.e., $|\boldsymbol{p}_k^\top \nabla f(\boldsymbol{\Sigma}_k + \alpha_k \boldsymbol{p}_k)| \leq c_2 |\boldsymbol{p}_k^\top \nabla f(\boldsymbol{\Sigma}_k)|$.

2.3. Riemannian Notations

Consider a Riemannian manifold denoted by $\mathcal{M}$. At every point $\boldsymbol{\Sigma} \in \mathcal{M}$, there is a tangent space to the manifold, denoted by $T_{\boldsymbol{\Sigma}}\mathcal{M}$. A tangent space includes tangent vectors. We denote a tangent vector by $\boldsymbol{\xi}$. For $\boldsymbol{\xi}, \boldsymbol{\eta} \in T_{\boldsymbol{\Sigma}}\mathcal{M}$, a metric on this manifold is the inner product defined on manifold and is denoted by $g_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}, \boldsymbol{\eta})$. Note that the gradient of a cost function, whose domain is a manifold, is a tangent vector in the tangent space and is denoted by $\nabla f(\boldsymbol{\Sigma})$ for point $\boldsymbol{\Sigma} \in \mathcal{M}$. Vector transport is an operator which maps a tangent vector $\boldsymbol{\xi} \in T_{\boldsymbol{\Sigma}_1}\mathcal{M}$ from its tangent space at point $\boldsymbol{\Sigma}_1$ to another tangent space at another point $\boldsymbol{\Sigma}_2$. We denote this vector transport by $\mathcal{T}_{\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2}(\boldsymbol{\xi}) : T_{\boldsymbol{\Sigma}_1}\mathcal{M} \mapsto T_{\boldsymbol{\Sigma}_2}\mathcal{M}$. Now, consider a point $\boldsymbol{\Sigma}$ on a manifold $\mathcal{M}$ and a descent direction $\boldsymbol{\xi}$ in the tangent space $T_{\boldsymbol{\Sigma}}\mathcal{M}$. The retraction $\mathrm{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}) : T_{\boldsymbol{\Sigma}}\mathcal{M} \mapsto \mathcal{M}$ retracts or maps the direction $\boldsymbol{\xi}$ in the tangent space onto the manifold $\mathcal{M}$. The operator exponential map, denoted by $\mathrm{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi})$, is also capable of this mapping by moving along the geodesic. One can use the second-order Taylor expansion of exponential map, which is positive-preserving (Jeuris et al., 2012) for the case of SPD manifold, for approximating the exponential map. In this paper, $\mathbf{tr}(.)$ denotes the trace of matrix.

2.4. Riemannian BFGS and LBFGS

The Riemannian extension of Euclidean BFGS (RBFGS) (Qi et al., 2010; Ring and Wirth, 2012; Huang et al., 2015) performs updates of Hessian approximation by Eq. (2) using Riemannian operators. RBFGS methods check both Wolfe conditions, Eqs. (4) and (5), which are computationally expensive. Cautious RBFGS (Huang et al., 2016) ignores the curvature condition and only checks the Armijo condition for linesearch. However, for ensuring that the Hessian approximation remains positive definite, it checks a cautious condition (Li and Fukushima, 2001) before updating the Hessian approximation.

Riemannian LBFGS (Sra and Hosseini, 2015, 2016; Hosseini and Sra, 2020) performs the recursions of LBFGS in the Riemannian space for finding the descent direction $\boldsymbol{\xi}_k \in T_{\boldsymbol{\Sigma}_k}\mathcal{M}$ and checks both Wolfe conditions in linesearch. In every iteration k of optimization, recursion starts with the direction $\boldsymbol{p} = -\nabla f(\boldsymbol{\Sigma}_k)$. Let the recursive function GetDirection$(\boldsymbol{p}, k)$ returns the descent direction. Inside every step of this recursion, we have (Hosseini and Sra, 2020, Algorithm 3):

$$\tilde{\boldsymbol{p}}_k := \boldsymbol{p}_k - \frac{\boldsymbol{g}_{\boldsymbol{\Sigma}_k}(\boldsymbol{s}_k, \boldsymbol{p}_k)}{\boldsymbol{g}_{\boldsymbol{\Sigma}_k}(\boldsymbol{y}_k, \boldsymbol{s}_k)}\boldsymbol{y}_k, \tag{6}$$

$$\widehat{\boldsymbol{p}}_k := \mathcal{T}_{\boldsymbol{\Sigma}_{k-1},\boldsymbol{\Sigma}_k}\big(\text{GetDirection}(\mathcal{T}^*_{\boldsymbol{\Sigma}_{k-1},\boldsymbol{\Sigma}_k}(\tilde{\boldsymbol{p}}), k-1)\big), \tag{7}$$

$$\text{return } \boldsymbol{\xi}_k := \widehat{\boldsymbol{p}}_k - \frac{\boldsymbol{g}_{\boldsymbol{\Sigma}_k}(\boldsymbol{y}_k, \widehat{\boldsymbol{p}}_k)}{\boldsymbol{g}_{\boldsymbol{\Sigma}_k}(\boldsymbol{y}_k, \boldsymbol{s}_k)}\boldsymbol{s}_k + \frac{\boldsymbol{g}_{\boldsymbol{\Sigma}_k}(\boldsymbol{s}_k, \boldsymbol{s}_k)}{\boldsymbol{g}_{\boldsymbol{\Sigma}_k}(\boldsymbol{y}_k, \boldsymbol{s}_k)}\boldsymbol{p}_k, \tag{8}$$

where, inspired by the introduced $\boldsymbol{s}_k$ and $\boldsymbol{y}_k$ for Euclidean spaces, we have:

$$\boldsymbol{\Sigma}_{k+1} := \text{Exp}_{\boldsymbol{\Sigma}_k}(\alpha_k\boldsymbol{\xi}_k) \text{ or } \text{Ret}_{\boldsymbol{\Sigma}_k}(\alpha_k\boldsymbol{\xi}_k), \tag{9}$$

$$\boldsymbol{s}_{k+1} := \mathcal{T}_{\boldsymbol{\Sigma}_k,\boldsymbol{\Sigma}_{k+1}}(\alpha_k\boldsymbol{\xi}_k), \tag{10}$$

$$\boldsymbol{y}_{k+1} := \nabla f(\boldsymbol{\Sigma}_{k+1}) - \mathcal{T}_{\boldsymbol{\Sigma}_k,\boldsymbol{\Sigma}_{k+1}}\big(\nabla f(\boldsymbol{\Sigma}_k)\big). \tag{11}$$

Note that the new point in every iteration is found by retraction, or an exponential map, of the searched point along the descent direction onto manifold. According to Eq. (7) in recursion and Eq. (10), RLBFGS involves both adjoint vector transport and vector transport which are computationally expensive. Moreover, Eqs. (6) and (8) show that Riemannian metric is utilized many times inside recursions. Our proposed mappings in this paper simply all vector transport, adjoint vector transport, and metric which are used in the RLBFGS algorithm.

2.5. Symmetric Positive Definite (SPD) Manifold

Consider the SPD manifold (Sra and Hosseini, 2015, 2016) whose every point is a SPD matrix, i.e., $\boldsymbol{\Sigma} \in \mathcal{M}$ and $\mathbb{S}^n_{++} \ni \boldsymbol{\Sigma} \succ \mathbf{0}$. It can be shown that the tangent space to the SPD manifold is the space of symmetric matrices, i.e., $T_{\mathcal{M}}(\boldsymbol{\Sigma}) \subset \mathbb{S}^n_{++}$ (Bhatia, 2009). In this paper, we focus on SPD manifolds which are widely used in data science. The

Table 1: Operators on SPD manifold under the proposed mappings.

Operator	No mapping
Metric, $g_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}, \boldsymbol{\eta})$	$\mathbf{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\eta})$
Gradient, $\nabla f(\boldsymbol{\Sigma})$	$\frac{1}{2}\boldsymbol{\Sigma}\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{\Sigma}$
Exponential map, $\mathrm{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi})$	$\boldsymbol{\Sigma}\exp(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}) = \boldsymbol{\Sigma}^{\frac{1}{2}}\exp(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-\frac{1}{2}})\,\boldsymbol{\Sigma}^{\frac{1}{2}}$
Vector transport, $\mathcal{T}_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi})$	$\boldsymbol{\Sigma}_2^{\frac{1}{2}}\boldsymbol{\Sigma}_1^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}_1^{-\frac{1}{2}}\boldsymbol{\Sigma}_2^{\frac{1}{2}}$ or $\boldsymbol{L}_2\boldsymbol{L}_1^{-1}\boldsymbol{\xi}\boldsymbol{L}_1^{-\top}\boldsymbol{L}_2^\top$
Approx. Euclidean retraction, $\mathrm{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi})$	$\boldsymbol{\Sigma} + \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}$

Operator	Mapping by inverse second root
Mapping	$\boldsymbol{\xi}' := \boldsymbol{\Sigma}^{-\frac{1}{2}}\,\boldsymbol{\xi}\,\boldsymbol{\Sigma}^{-\frac{1}{2}}$
Metric, $g'_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}', \boldsymbol{\eta}')$	$\mathbf{tr}(\boldsymbol{\xi}'\boldsymbol{\eta}')$
Gradient, $\nabla' f(\boldsymbol{\Sigma})$	$\frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{\Sigma}^{\frac{1}{2}}$
Exponential map, $\mathrm{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}')$	$\boldsymbol{\Sigma}^{\frac{1}{2}}\exp(\boldsymbol{\xi}')\,\boldsymbol{\Sigma}^{\frac{1}{2}}$
Vector transport, $\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi}')$	$\boldsymbol{\xi}'$
Approx. Euclidean retraction, $\mathrm{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}')$	$\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'\boldsymbol{\Sigma}^{\frac{1}{2}} + \frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'^2\boldsymbol{\Sigma}^{\frac{1}{2}}$

Operator	Mapping by Cholesky decomposition
Mapping	$\boldsymbol{\xi}' := \boldsymbol{L}^{-1}\,\boldsymbol{\xi}\,\boldsymbol{L}^{-\top}$
Metric, $g'_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}', \boldsymbol{\eta}')$	$\mathbf{tr}(\boldsymbol{\xi}'\boldsymbol{\eta}')$
Gradient, $\nabla' f(\boldsymbol{\Sigma})$	$\frac{1}{2}\boldsymbol{L}^\top\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{L}$
Exponential map, $\mathrm{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}')$	$\boldsymbol{\Sigma}\exp(\boldsymbol{L}^{-\top}\boldsymbol{\xi}'\boldsymbol{L}^\top)$
Vector transport, $\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi}')$	$\boldsymbol{\xi}'$
Approx. Euclidean retraction, $\mathrm{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}')$	$\boldsymbol{\Sigma} + \boldsymbol{L}\,\boldsymbol{\xi}'\boldsymbol{L}^\top + \frac{1}{2}\boldsymbol{L}\,\boldsymbol{\xi}'^2\boldsymbol{L}^\top$

operators for metric, gradient, exponential map, and vector transport on SPD manifolds are listed in Table 1 (Sra and Hosseini, 2016; Hosseini and Sra, 2020). In this table, $\nabla_E f(\boldsymbol{\Sigma})$ denotes the Euclidean gradient and $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ are the lower-triangular matrices in Cholesky decomposition of points $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$, respectively.

3. Vector Transport Free Riemannian LBFGS Using Mapping by Inverse Second Root

We propose two mappings on tangent vectors in the tangent space where the first mapping is by inverse second root. Our first proposed mapping in the tangent space of every point $\boldsymbol{\Sigma} \in \mathcal{M}$ is:

$$\boldsymbol{\xi}' := \boldsymbol{\Sigma}^{-\frac{1}{2}}\,\boldsymbol{\xi}\,\boldsymbol{\Sigma}^{-\frac{1}{2}} \implies \boldsymbol{\xi} = \boldsymbol{\Sigma}^{\frac{1}{2}}\,\boldsymbol{\xi}'\,\boldsymbol{\Sigma}^{\frac{1}{2}}, \tag{12}$$

where the mapped tangent vector still remains in the tangent space, i.e. $\boldsymbol{\xi}, \boldsymbol{\xi}' \in T_{\boldsymbol{\Sigma}}\mathcal{M} \subset \mathbb{S}_{++}^n$. It is important the proposed mapping is bijective and keeps the tangent vector in the tangent space while it simplifies vector transport, adjoint vector transport, and metric.

Under the proposed mapping (12), the Riemannian operators on a SPD manifold are modified and mostly simplified. These operators are listed in Table 1. In the following, we provide proofs for these modifications.

Proposition 1 *After mapping (12), the metric on SPD manifold is reduced to the Euclidean inner product, i.e.,*

$$g_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}', \boldsymbol{\eta}') = \boldsymbol{tr}(\boldsymbol{\xi}'\boldsymbol{\eta}'). \tag{13}$$

Proof

$$g_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}', \boldsymbol{\eta}') \overset{(a)}{=} \mathbf{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\eta}) = \mathbf{tr}(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\eta})$$
$$\overset{(b)}{=} \mathbf{tr}(\underbrace{\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-\frac{1}{2}}}_{=\boldsymbol{\xi}'}\underbrace{\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\eta}\boldsymbol{\Sigma}^{-\frac{1}{2}}}_{=\boldsymbol{\eta}'}) \overset{(12)}{=} \mathbf{tr}(\boldsymbol{\xi}'\boldsymbol{\eta}'),$$

where (a) is because of definition of metric on SPD manifolds (see Table 1) and (b) is thanks to the cyclic property of trace. Q.E.D. ■

Proposition 2 *After mapping (12), the gradient on a SPD manifold is changed to:*

$$\nabla' f(\boldsymbol{\Sigma}) = \frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{\Sigma}^{\frac{1}{2}}, \tag{14}$$

where $\nabla_E f(\boldsymbol{\Sigma})$ denotes the Euclidean gradient.

Proof

$$\nabla' f(\boldsymbol{\Sigma}) \overset{(12)}{=} \boldsymbol{\Sigma}^{-\frac{1}{2}}\nabla f(\boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-\frac{1}{2}} \overset{(a)}{=} \frac{1}{2}\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-\frac{1}{2}}$$
$$= \frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{\Sigma}^{\frac{1}{2}},$$

where (a) is due to definition of gradient on a SPD manifold (see Table 1). Q.E.D. ■

Proposition 3 *After mapping (12), the vector transport is changed to identity, i.e.,*

$$\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi}') = \boldsymbol{\xi}', \tag{15}$$

hence, optimization becomes vector transport free.

Proof

$$\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi}') \overset{(12)}{=} \boldsymbol{\Sigma}_2^{-\frac{1}{2}}\mathcal{T}_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi})\boldsymbol{\Sigma}_2^{-\frac{1}{2}} \overset{(a)}{=} \underbrace{\boldsymbol{\Sigma}_2^{-\frac{1}{2}}(\boldsymbol{\Sigma}_2^{\frac{1}{2}}}_{=\boldsymbol{I}}\boldsymbol{\Sigma}_1^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}_1^{-\frac{1}{2}}\underbrace{\boldsymbol{\Sigma}_2^{\frac{1}{2}})\boldsymbol{\Sigma}_2^{-\frac{1}{2}}}_{=\boldsymbol{I}} = \boldsymbol{\Sigma}_1^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}_1^{-\frac{1}{2}} \overset{(12)}{=} \boldsymbol{\xi}',$$

where (a) is due to definition of vector transport on a SPD manifold (see Table 1). Q.E.D. ■

Proposition 4 *After mapping (12), the exponential map on a SPD manifold becomes:*

$$\textit{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') = \boldsymbol{\Sigma}^{\frac{1}{2}}\, \textit{exp}(\boldsymbol{\xi}')\, \boldsymbol{\Sigma}^{\frac{1}{2}}. \tag{16}$$

Proof

$$\mathrm{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') \overset{(a)}{=} \boldsymbol{\Sigma} \exp(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}) \overset{(b)}{=} \boldsymbol{\Sigma}^{\frac{1}{2}} \exp(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-\frac{1}{2}})\, \boldsymbol{\Sigma}^{\frac{1}{2}} \overset{(12)}{=} \boldsymbol{\Sigma}^{\frac{1}{2}} \exp(\boldsymbol{\xi}')\, \boldsymbol{\Sigma}^{\frac{1}{2}},$$

where (a) is shown in (Sra and Hosseini, 2015, Eq. 3.3) and (b) is shown in (Sra and Hosseini, 2015, Eq. 3.2). Also see Table 1. Q.E.D. ■

Proposition 5 *After mapping (12), the adjoint of vector transport on a SPD manifold remains the same. In other words, if before mapping we have the definition of adjoint vector transport as (Ring and Wirth, 2012):*

$$g_{\boldsymbol{\Sigma}_1}(\boldsymbol{\xi}, \mathcal{T}^*_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}\boldsymbol{\eta}) = g_{\boldsymbol{\Sigma}_2}(\mathcal{T}_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}\boldsymbol{\xi}, \boldsymbol{\eta}), \tag{17}$$

$\forall \boldsymbol{\xi} \in T_{\boldsymbol{\Sigma}_1}\mathcal{M}, \forall \boldsymbol{\eta} \in T_{\boldsymbol{\Sigma}_2}\mathcal{M}$, *we will have:*

$$g_{\boldsymbol{\Sigma}_1}(\boldsymbol{\xi}', \mathcal{T}'^*_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}\boldsymbol{\eta}') = g_{\boldsymbol{\Sigma}_2}(\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}\boldsymbol{\xi}', \boldsymbol{\eta}'), \tag{18}$$

$\forall \boldsymbol{\xi}' \in T_{\boldsymbol{\Sigma}_1}\mathcal{M}, \forall \boldsymbol{\eta}' \in T_{\boldsymbol{\Sigma}_2}\mathcal{M}$.

Corollary 6 *It can be concluded directly from the definition, as well as Propositions 1, 3, and 5 that the adjoint of vector transport is again identity under mapping (12).*

Proposition 7 *After mapping (12), the approximation of Euclidean retraction, using second-order Taylor expansion, on a SPD manifold becomes:*

$$\textit{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') = \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'\boldsymbol{\Sigma}^{\frac{1}{2}} + \frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'^2\boldsymbol{\Sigma}^{\frac{1}{2}}. \tag{19}$$

Proof

$$\begin{aligned}\mathrm{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') &\overset{(a)}{=} \boldsymbol{\Sigma} + \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi} \overset{(12)}{=} \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\frac{1}{2}}\,\boldsymbol{\xi}'\,\boldsymbol{\Sigma}^{\frac{1}{2}} + \frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'\,\underbrace{\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{\frac{1}{2}}}_{=\boldsymbol{I}}\,\boldsymbol{\xi}'\boldsymbol{\Sigma}^{\frac{1}{2}} \\ &= \boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'\boldsymbol{\Sigma}^{\frac{1}{2}} + \frac{1}{2}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\xi}'^2\boldsymbol{\Sigma}^{\frac{1}{2}},\end{aligned}$$

where (a) is because of approximation of Euclidean retraction, using the second-order Taylor expansion, on SPD manifolds (see Table 1). Q.E.D. ■

4. Vector Transport Free Riemannian LBFGS Using Mapping by Cholesky Decomposition

Our second proposed mapping is by Cholesky decomposition which is very efficient computationally. Consider the Cholesky decomposition of point $\boldsymbol{\Sigma} \in \mathcal{M}$ (Golub and Van Loan, 2013):

$$\mathbf{0} \prec \boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^\top \implies \boldsymbol{\Sigma}^{-1} = \boldsymbol{L}^{-\top}\boldsymbol{L}^{-1}, \tag{20}$$

where $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ is the lower-triangular matrix in Cholesky decomposition. It is noteworthy that many of the MATLAB matrix multiplication operators, which the Manopt toolbox (Boumal et al., 2014) also uses, apply Cholesky decomposition internally due to its efficiency.

In the tangent space of every point $\boldsymbol{\Sigma} \in \mathcal{M}$, the proposed mapping is:

$$\boldsymbol{\xi}' := \boldsymbol{L}^{-1}\,\boldsymbol{\xi}\,\boldsymbol{L}^{-\top} \implies \boldsymbol{\xi} = \boldsymbol{L}\,\boldsymbol{\xi}'\,\boldsymbol{L}^\top, \tag{21}$$

where $\boldsymbol{\xi}, \boldsymbol{\xi}' \in T_{\boldsymbol{\Sigma}}\mathcal{M} \subset \mathbb{S}^n_{++}$. Note that, similar to the previous mapping, the tangent matrix is still symmetric under this mapping; hence, it remains in the tangent space of the SPD manifold (Bhatia, 2009). Similar to the previous mapping, under the second proposed mapping (12), the Riemannian operators on SPD manifold are simplified. These operators can be found in Table 1. In the following, we provide proofs for these operators.

Proposition 8 *After mapping (21), the metric on a SPD manifold is reduced to the Euclidean inner product, i.e.,*

$$g_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}', \boldsymbol{\eta}') = \boldsymbol{tr}(\boldsymbol{\xi}'\boldsymbol{\eta}'). \tag{22}$$

Proof

$$\begin{aligned} g_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}', \boldsymbol{\eta}') &\overset{(a)}{=} \mathbf{tr}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\eta}) = \mathbf{tr}(\boldsymbol{L}^{-\top}\boldsymbol{L}^{-1}\boldsymbol{\xi}\boldsymbol{L}^{-\top}\boldsymbol{L}^{-1}\boldsymbol{\eta}) \\ &\overset{(b)}{=} \mathbf{tr}(\underbrace{\boldsymbol{L}^{-1}\boldsymbol{\xi}\boldsymbol{L}^{-\top}}_{=\boldsymbol{\xi}'}\underbrace{\boldsymbol{L}^{-1}\boldsymbol{\eta}\boldsymbol{L}^{-\top}}_{=\boldsymbol{\eta}'}) \overset{(21)}{=} \mathbf{tr}(\boldsymbol{\xi}'\boldsymbol{\eta}'), \end{aligned}$$

where (a) is because of definition of metric on SPD manifolds (see Table 1) and (b) is thanks to the cyclic property of trace. Q.E.D. ■

Proposition 9 *After mapping (21), the gradient on SPD manifold is changed to:*

$$\nabla' f(\boldsymbol{\Sigma}) = \frac{1}{2}\boldsymbol{L}^\top\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{L}, \tag{23}$$

where $\nabla_E f(\boldsymbol{\Sigma})$ denotes the Euclidean gradient.

Proof

$$\begin{aligned} \nabla' f(\boldsymbol{\Sigma}) &\overset{(21)}{=} \boldsymbol{L}^{-1}\nabla f(\boldsymbol{\Sigma})\boldsymbol{L}^{-\top} \overset{(a)}{=} \frac{1}{2}\boldsymbol{L}^{-1}\boldsymbol{\Sigma}\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{\Sigma}\boldsymbol{L}^{-\top} \\ &\overset{(20)}{=} \frac{1}{2}\boldsymbol{L}^{-1}\boldsymbol{L}\boldsymbol{L}^\top\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{L}^{-\top} \\ &= \frac{1}{2}\boldsymbol{L}^\top\big(\nabla_E f(\boldsymbol{\Sigma}) + (\nabla_E f(\boldsymbol{\Sigma}))^\top\big)\boldsymbol{L}, \end{aligned}$$

where (a) is due to definition of gradient on a SPD manifold (see Table 1). Q.E.D. ■

Proposition 10 *After mapping (21), the vector transport is changed to identity, i.e.,*

$$\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi}') = \boldsymbol{\xi}', \tag{24}$$

hence, optimization becomes vector transport free.

Proof

$$\mathcal{T}'_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi}') \overset{(21)}{=} \boldsymbol{L}_2^{-1}\mathcal{T}_{\boldsymbol{\Sigma}_1,\boldsymbol{\Sigma}_2}(\boldsymbol{\xi})\boldsymbol{L}_2^{-\top} \overset{(a)}{=} \underbrace{\boldsymbol{L}_2^{-1}(\boldsymbol{L}_2}_{=\boldsymbol{I}}\boldsymbol{L}_1^{-1}\boldsymbol{\xi}\boldsymbol{L}_1^{-\top}\underbrace{\boldsymbol{L}_2^{\top})\boldsymbol{L}_2^{-\top}}_{=\boldsymbol{I}} = \boldsymbol{L}_1^{-1}\boldsymbol{\xi}\boldsymbol{L}_1^{-\top} \overset{(21)}{=} \boldsymbol{\xi}',$$

where (a) is due to definition of vector transport on SPD manifold (see Table 1). Q.E.D. ■

Proposition 11 *After mapping (21), the exponential map on SPD manifold becomes:*

$$Exp_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') = \boldsymbol{\Sigma}\, exp(\boldsymbol{L}^{-\top}\boldsymbol{\xi}'\boldsymbol{L}^{\top}). \tag{25}$$

Proof

$$\text{Exp}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') \overset{(a)}{=} \boldsymbol{\Sigma}\exp(\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi}) \overset{(b)}{=} \boldsymbol{\Sigma}\exp(\boldsymbol{L}^{-\top}\underbrace{\boldsymbol{L}^{-1}\boldsymbol{L}}_{=\boldsymbol{I}}\boldsymbol{\xi}'\boldsymbol{L}^{\top}) = \boldsymbol{\Sigma}\exp(\boldsymbol{L}^{-\top}\boldsymbol{\xi}'\boldsymbol{L}^{\top}),$$

where (a) is shown in (Sra and Hosseini, 2015, Eq. 3.3) and (b) is because of Eqs. (20) and (21). Q.E.D. ■

Similar to Proposition 5 and Corollary 6, the adjoint vector transport becomes identity under mapping (21).

Proposition 12 *After mapping (21), the approximation of Euclidean retraction by second-order Taylor expansion on a SPD manifold becomes:*

$$Ret_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') = \boldsymbol{\Sigma} + \boldsymbol{L}\,\boldsymbol{\xi}'\boldsymbol{L}^{\top} + \frac{1}{2}\boldsymbol{L}\,\boldsymbol{\xi}'^{2}\boldsymbol{L}^{\top}. \tag{26}$$

Proof

$$\text{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') \overset{(a)}{=} \boldsymbol{\Sigma} + \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}\boldsymbol{\Sigma}^{-1}\boldsymbol{\xi} \overset{(b)}{=} \boldsymbol{\Sigma} + \boldsymbol{L}\,\boldsymbol{\xi}'\boldsymbol{L}^{\top} + \frac{1}{2}\boldsymbol{L}\,\boldsymbol{\xi}'\underbrace{\boldsymbol{L}^{\top}\boldsymbol{L}^{-\top}}_{=\boldsymbol{I}}\underbrace{\boldsymbol{L}^{-1}\boldsymbol{L}}_{=\boldsymbol{I}}\boldsymbol{\xi}'\boldsymbol{L}^{\top},$$

which gives Eq. (26), where (a) is because of approximation of Euclidean retraction, using second-order Taylor expansion, on SPD manifolds (see Table 1), and (b) is because of Eqs. (20) and (21). Q.E.D. ■

Noticing Eq. (21), the approximation of retraction under mapping by Cholesky decomposition, i.e. Eq. (26), can be restated as $\text{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') = 0.5\,\boldsymbol{\Sigma} + 0.5\,\boldsymbol{L}(\boldsymbol{I}+\boldsymbol{\xi}')^2\boldsymbol{L}^{\top}$. Defining $\boldsymbol{\Psi} := \boldsymbol{L}(\boldsymbol{I}+\boldsymbol{\xi}')$ restates this retraction as $\text{Ret}_{\boldsymbol{\Sigma}}(\boldsymbol{\xi}') = 0.5\,\boldsymbol{\Sigma} + 0.5\,\boldsymbol{\Psi}\boldsymbol{\Psi}^{\top}$ because $\boldsymbol{\xi}'$ is symmetric. The term $\boldsymbol{\Psi}\boldsymbol{\Psi}^{\top}$ is very efficient and fast to compute because it is symmetric.

Corollary 13 *Following Proposition 3, Corollary 6, and Proposition 10, we see that under mapping (12) or (21), both vector transport and adjoint vector transport are identity. As these transforms become identity, they also become isometric because inner products of vectors do not change under these transforms. As they are identity, these transforms also preserve the length of vectors under the proposed mappings.*

Propositions 1 and 8 and Corollary 13 show the two proposed mappings simplify vector transport and adjoint vector transport to isometric identity and reduce the Riemannian metric to Euclidean inner product. These reductions and simplifications reduce computations significantly during optimization on the manifold.

5. Simulations

In this section, we evaluate the effectiveness of the proposed mappings, i.e. (12) and (21), in the tangent space. Here, we show that these mappings often improve the performance and speed of RLBFGS. The code of this article will be posted online on GitHub after acceptance.

5.1. Optimization Problem

The optimization problem, which we selected for evaluation, is the Riemannian optimization for Gaussian Mixture Model (GMM) without the use of expectation maximization. We employ RLBFGS with and without the proposed mappings for fitting the GMM problem whose algorithm can be found in (Hosseini and Sra, 2020; Hosseini and Mash'al, 2015). This is a suitable problem for evaluation of the proposed mappings because the covariance matrices are SPD (Bhatia, 2009). For this, we minimize the negative log-likelihood of GMM where the covariance matrix is constrained to belong to the SPD matrix manifold (Hosseini and Sra, 2020). For n-dimensional GMM, the optimization problem is:

$$\begin{aligned} \underset{\{\alpha_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\}_{j=1}^K}{\text{minimize}} \quad & -\sum_{i=1}^{N} \log\Big(\sum_{j=1}^{K} \alpha_j\, \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)\Big), \\ \text{subject to} \quad & \boldsymbol{\Sigma}_j \in \mathcal{M} = \mathbb{S}_{++}^n, \quad \forall j \in \{1, \dots, K\}, \end{aligned} \tag{27}$$

where N denotes the sample size, K denotes the number of components in mixture model, and α_j, $\boldsymbol{\mu}_j$, and $\boldsymbol{\Sigma}_j$ are the mixing probability, mean, and covariance of the j-th component, respectively. We use the same reformulation trick of (Hosseini and Sra, 2020) to reformulate the cost function of (27).

5.2. Experimental Setup

We solved the optimization problem (27) using RLBFGS which is focused in this paper. For all experiments, with and without the proposed mappings, we use both Wolfe linesearch conditions. The mixture parameters were initialized using K-means++ (Arthur and Vassilvitskii, 2007) following (Hosseini and Sra, 2020). Three different levels of separation of Gaussian models, namely low, mid, and high, were used. The reader can refer to (Hosseini and Sra, 2020) for mathematical details of these separation levels. For every experiment, we performed optimization for ten runs and the reported results are the average of performances over the runs. In our reports, we denote the proposed Vector Transport Free (VTF) RLBFGS with VTF-RLBFGS where ISR and Cholesky (or Chol.) stand for VTF-RLBFGS using mapping by inverse second root and Cholesky decomposition, respectively. The programming language, used for experiments, was MATLAB and the hardware was Intel Core-i7 CPU with the base frequency 2.20 GHz and 12 GB RAM.

5.3. Evaluation of the Proposed Mappings

The average number of iterations, convergence time, time per iteration, and the cost value of last iteration are reported in Tables 2 and 3 where exponential map is used in former and Taylor approximation of exponential map is used for retraction in the latter. In these experiments, we report performances for $K \in \{2, 5\}$, $n \in \{2, 10\}$, $N = 10n^2 \in \{40, 1000\}$. For the sake of fairness, all the three compared algorithms start with the same initial points in every run. The reader can see more extensive experiments for more sample size and dimensionality, i.e. $K \in \{2, 5\}$, $n \in \{2, 10, 100\}$, $N = 100n^2 \in \{400, 10000, 1000000\}$, in the Supplementary Material.

5.3.1. Discussion by Varying Separability

As Tables 2 and 3 show, the proposed ISR mapping converges faster than no mapping most often in all separability levels. Both its time per iteration and number of iterations are often less than no mapping. These tables also show that the proposed Cholesky mapping converges faster than no mapping in most of the cases, although not all cases. Overall, we see that the two proposed mappings make RLBFGS faster and more efficient most often. This pacing improvement can be noticed more for low and mid separability levels because they are harder cases to converge.

In terms of quality of local minimum, the proposed mappings often find the same local minimum as no mapping, but in a faster way. The equality of the found local optima in the three methods is because of using the same RLBFGS algorithm as their base in addition to having the same initial points. In some cases, the proposed mappings have even found better local minima.

5.3.2. Discussion by Varying Dimensionality

We can discuss the results of Tables 2 and 3 by varying dimensionality, $n \in \{2, 10\}$. Tables 1 and 2 in Supplementary Material also report performance for dimensions $n \in \{2, 10, 100\}$ for larger sample size. Obviously, by increasing dimensionality, the time of convergence goes up and the faster pacing of the proposed mappings is noticed more compared to no mapping.

5.3.3. Discussion by Varying the Number of Components

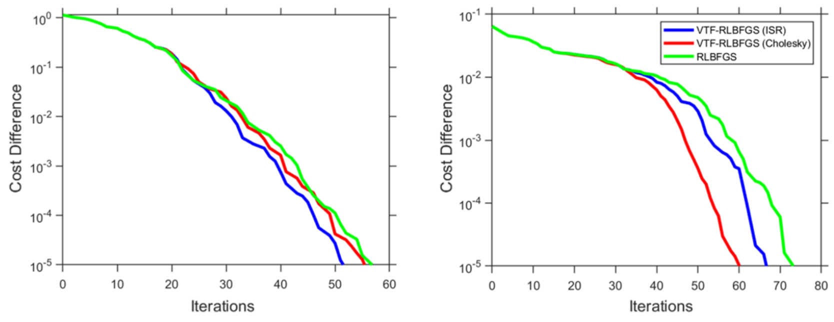
The results of Tables 2 and 3 can also be interpreted based on varying the number of mixture components, $K \in \{2, 5\}$. The more number of components makes the optimization problem harder. Hence, the difference of speeds of the proposed mappings and no mapping can be noticed more for $K = 5$; although, for both K values, the proposed mappings are often faster than no mapping.

Table 2: Comparison of average results over ten runs where exponential map is used in algorithms and $K \in \{2, 5\}$, $n \in \{2, 10\}$, $N = 10n^2 \in \{40, 1000\}$.

K	n	Separation	Algorithm	#iters	conv. time	iter. time	last cost
2	2	Low	VTF (ISR)	53.100±18.248	68.380±52.834	1.140±0.504	0.364±0.444
			VTF (Chol.)	52.100±17.866	61.096±48.433	1.030±0.463	0.364±0.444
			RLBFGS	52.500±16.595	65.890±49.208	1.125±0.458	0.364±0.444
		Mid	VTF (ISR)	56.400±21.813	76.124±69.189	1.150±0.574	0.657±0.344
			VTF (Chol.)	54.400±19.156	68.456±64.290	1.099±0.504	0.638±0.333
			RLBFGS	57.700±19.844	81.957±64.463	1.250±0.550	0.657±0.344
		High	VTF (ISR)	25.500±3.064	9.518±2.922	0.366±0.067	0.341±0.371
			VTF (Chol.)	26.000±4.738	10.254±5.468	0.377±0.108	0.341±0.371
			RLBFGS	25.500±3.064	10.054±3.141	0.386±0.073	0.341±0.371
	10	Low	VTF (ISR)	77.600±54.175	235.615±466.119	1.936±1.751	4.210±0.889
			VTF (Chol.)	84.100±73.260	313.398±714.908	2.041±2.150	4.209±0.889
			RLBFGS	77.600±52.754	242.743±458.641	2.069±1.733	4.209±0.889
		Mid	VTF (ISR)	44.600±7.260	43.654±16.872	0.948±0.208	4.262±1.098
			VTF (Chol.)	45.900±8.647	45.661±20.088	0.955±0.236	4.262±1.098
			RLBFGS	45.200±8.080	48.104±19.981	1.025±0.243	4.262±1.098
		High	VTF (ISR)	44.400±9.252	43.684±22.460	0.936±0.254	3.874±1.395
			VTF (Chol.)	47.100±8.333	48.278±21.112	0.987±0.240	3.874±1.395
			RLBFGS	43.300±7.150	43.904±17.626	0.981±0.225	3.874±1.395
5	2	Low	VTF (ISR)	152.400±62.819	1344.573±999.949	7.560±3.406	0.260±0.458
			VTF (Chol.)	174.800±114.139	2168.886±3090.820	8.680±6.348	0.265±0.466
			RLBFGS	166.300±76.884	1767.676±1406.454	8.788±4.427	0.267±0.454
		Mid	VTF (ISR)	120.700±69.620	942.713±1063.075	5.807±3.877	0.781±0.207
			VTF (Chol.)	121.600±65.030	897.110±988.826	5.720±3.445	0.781±0.207
			RLBFGS	136.300±91.493	1404.654±1986.798	7.057±5.378	0.764±0.225
		High	VTF (ISR)	46.400±17.322	94.741±105.045	1.739±0.902	1.805±0.385
			VTF (Chol.)	46.200±19.803	98.065±122.950	1.729±1.020	1.805±0.385
			RLBFGS	46.200±18.937	104.934±126.734	1.879±1.064	1.805±0.385
	10	Low	VTF (ISR)	295.900±86.577	5524.495±3173.855	17.299±5.221	6.175±0.745
			VTF (Chol.)	292.300±83.828	5294.565±3165.735	16.737±5.350	6.197±0.718
			RLBFGS	318.800±100.216	7083.082±4801.718	20.279±6.859	6.173±0.744
		Mid	VTF (ISR)	134.200±54.956	1139.332±919.917	7.302±3.227	6.753±0.708
			VTF (Chol.)	133.300±52.415	1100.519±842.840	7.183±3.045	6.753±0.708
			RLBFGS	135.300±54.965	1268.707±967.678	8.070±3.580	6.753±0.708
		High	VTF (ISR)	68.600±12.367	241.487±87.593	3.398±0.764	6.599±0.836
			VTF (Chol.)	74.000±14.071	279.271±105.828	3.632±0.838	6.599±0.836
			RLBFGS	68.300±11.982	258.475±93.959	3.661±0.790	6.599±0.836

Table 3: Comparison of average results over ten runs where retraction with Taylor series expansion is used in algorithms and $K \in \{2, 5\}$, $n \in \{2, 10\}$, $N = 10n^2 \in \{40, 1000\}$.

K	n	Separation	Algorithm	#iters	conv. time	iter. time	last cost
2	2	Low	VTF (ISR)	59.800±22.553	93.135±88.437	1.320±0.711	0.610±0.366
			VTF (Chol.)	62.600±29.079	106.545±117.633	1.355±0.830	0.619±0.382
			RLBFGS	59.800±26.803	100.424±120.310	1.363±0.787	0.610±0.366
		Mid	VTF (ISR)	45.800±16.982	48.155±38.045	0.900±0.454	0.482±0.497
			VTF (Chol.)	47.000±18.074	51.529±41.465	0.930±0.483	0.482±0.497
			RLBFGS	45.000±16.330	48.150±37.285	0.921±0.458	0.482±0.497
		High	VTF (ISR)	25.600±4.142	9.816±3.837	0.370±0.093	0.270±0.456
			VTF (Chol.)	26.300±4.448	10.540±4.190	0.385±0.101	0.270±0.456
			RLBFGS	25.500±4.353	10.526±4.819	0.396±0.113	0.270±0.456
	10	Low	VTF (ISR)	75.800±25.607	166.736±137.600	1.955±0.816	4.129±0.771
			VTF (Chol.)	74.500±22.629	152.074±107.531	1.855±0.682	4.129±0.771
			RLBFGS	74.800±25.442	172.641±143.198	2.054±0.834	4.129±0.771
		Mid	VTF (ISR)	50.900±13.956	64.501±38.055	1.170±0.395	3.472±1.154
			VTF (Chol.)	52.100±13.892	66.775±41.257	1.184±0.410	3.472±1.154
			RLBFGS	51.000±14.063	70.132±46.421	1.264±0.450	3.472±1.154
		High	VTF (ISR)	43.600±5.522	42.763±11.897	0.963±0.168	4.353±1.081
			VTF (Chol.)	47.400±6.620	50.438±15.187	1.041±0.182	4.353±1.081
			RLBFGS	44.300±6.464	47.830±14.747	1.055±0.192	4.353±1.081
5	2	Low	VTF (ISR)	262.500±126.532	4430.577±4455.877	13.849±6.991	0.082±0.281
			VTF (Chol.)	253.500±124.970	4086.076±3967.586	13.084±6.850	0.099±0.279
			RLBFGS	270.700±144.145	5305.821±5495.383	15.560±8.452	0.090±0.282
		Mid	VTF (ISR)	110.900±50.886	731.318±769.639	5.445±2.806	1.010±0.349
			VTF (Chol.)	112.200±60.736	792.756±1053.832	5.436±3.358	1.010±0.349
			RLBFGS	111.900±55.183	814.994±975.737	5.827±3.289	1.010±0.349
		High	VTF (ISR)	52.000±13.565	116.485±69.535	2.071±0.728	1.626±0.431
			VTF (Chol.)	51.400±12.616	114.032±67.916	2.057±0.735	1.626±0.431
			RLBFGS	53.400±14.081	139.989±89.213	2.408±0.936	1.626±0.431
	10	Low	VTF (ISR)	219.700±57.908	2946.040±1513.372	12.579±3.507	6.643±0.662
			VTF (Chol.)	226.100±86.010	3272.111±2808.634	12.707±5.156	6.625±0.650
			RLBFGS	231.300±64.484	3607.215±1947.580	14.516±4.305	6.642±0.663
		Mid	VTF (ISR)	87.100±21.502	413.616±223.510	4.458±1.310	6.585±0.569
			VTF (Chol.)	87.200±21.186	405.484±214.985	4.374±1.262	6.585±0.569
			RLBFGS	87.400±20.403	454.434±227.556	4.918±1.339	6.585±0.569
		High	VTF (ISR)	60.500±10.533	181.113±76.070	2.892±0.649	6.827±0.836
			VTF (Chol.)	62.700±11.295	197.565±93.653	3.019±0.827	6.827±0.836
			RLBFGS	58.900±11.040	187.518±86.254	3.058±0.746	6.827±0.836



(a) (b)

Figure 1: The cost difference progress for several runs: (a) $K = 2$, $n = 10$, $N = 1000$, mid separation, with exponential map, and (b) $K = 2$, $n = 2$, $N = 400$, mid separation, with exponential map.

5.3.4. Discussion by Varying Retraction Type

We can also compare the performance of algorithms in terms of type of retraction. Tables 2 and 3 report performances where exponential map and Taylor approximation of exponential map are used for retraction, respectively. Comparing these tables shows that using Taylor approximation usually converges faster than using exponential map which makes sense because exponential map requires passing on geodesics. This difference of pacing is more obvious for larger number of components, i.e., $K = 5$. Our two proposed mappings outperform no mapping for both types retraction. This shows that our mappings are effective regardless of the details of operators.

5.3.5. Discussion by Varying the Sample Size

Tables 2 and 3 report for $n \in \{2, 10\}$, $N = 10n^2 \in \{40, 1000\}$. More experiments for larger sample size and dimensionality, i.e. $K \in \{2, 5\}$, $n \in \{2, 10, 100\}$, $N = 100n^2 \in \{400, 10000, 1000000\}$, can be found in Tables 1 and 2 of Supplementary Material. Comparing those tables with Tables 2 and 3 shows that larger sample size and/or dimensionality takes more time to converge as expected. Still, our proposed mappings often converge faster than no mapping. The difference of pacing is mostly less in larger sample size compared to smaller sample size. This is because very large sample size consumes time on computation of cost function and the difference is not given much chance to show off in that case.

5.3.6. Discussion by Cost Difference Progress

The log-scale cost difference progress of several insightful runs of RLBFGS, with and without the proposed mappings, are illustrated in Fig. 1. A complete set of plots for cost difference progress can be seen in Figs. 1 to 4 of Supplementary Material. Figs. 1-a and 1-b show that in some cases, ISR mapping is faster than the Cholesky mapping and in some other cases, we have the other way around.

6. Conclusion and Future Direction

In this paper, we proposed two mappings in the tangent space of SPD manifolds by inverse second root and Cholesky decomposition. The proposed mappings simplify the vector transports and adjoint vector transports to identity. These transports are widely used in RLBFGS quasi-Newton optimization, to identity. They also reduce the Riemannian metric to the Euclidean inner product which is more efficient computationally. Simulation results verified the effectiveness of the proposed mappings for an optimization task on SPD matrix manifolds. In this work, we focused on mappings for SPD manifolds which are widely used in machine learning and data science. A possible future direction is to extend the proposed mappings for other well-known Riemannian manifolds, such as Grassmann and Stiefel (Edelman et al., 1998), as well as other Riemannian optimization methods. This paper opens a new research path for such mappings in the tangent space and we conjecture that such mappings can make numerical Riemannian optimization more efficient.

References

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

Rajendra Bhatia. *Positive definite matrices*. Princeton University Press, 2009.

Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Available online, 2020.

Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15 (1):1455–1459, 2014.

Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2): 303–353, 1998.

Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU press, 2013.

Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

Reshad Hosseini and Mohamadreza Mash'al. Mixest: An estimation toolbox for mixture models. *arXiv preprint arXiv:1507.06065*, 2015.

Reshad Hosseini and Suvrit Sra. An alternative to EM for Gaussian mixture models: batch and stochastic Riemannian optimization. *Mathematical Programming*, 181(1):187–223, 2020.

Wen Huang, Kyle A Gallivan, and P-A Absil. A Broyden class of quasi-Newton methods for Riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

Wen Huang, P-A Absil, and Kyle A Gallivan. A Riemannian BFGS method for nonconvex optimization problems. In *Numerical Mathematics and Advanced Applications ENUMATH 2015*, pages 627–634. Springer, 2016.

Ben Jeuris, Raf Vandebril, and Bart Vandereycken. A survey and comparison of contemporary algorithms for computing the matrix geometric mean. *Electronic Transactions on Numerical Analysis*, 39(ARTICLE):379–402, 2012.

Huibo Ji. *Optimization approaches on smooth manifolds.* PhD thesis, Australian National University, 2007.

Dong-Hui Li and Masao Fukushima. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11(4): 1054–1064, 2001.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Jorge Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

Jorge Nocedal and Stephen Wright. *Numerical optimization.* Springer Science & Business Media, 2006.

Chunhong Qi, Kyle A Gallivan, and P-A Absil. Riemannian BFGS algorithm with applications. In *Recent advances in optimization and its applications in engineering*, pages 183–192. Springer, 2010.

Wolfgang Ring and Benedikt Wirth. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, 2012.

Matthias Seibert, Martin Kleinsteuber, and Knut Hüper. Properties of the BFGS method on Riemannian manifolds. *Mathematical System Theory C Festschrift in Honor of Uwe Helmke on the Occasion of his Sixtieth Birthday*, pages 395–412, 2013.

Suvrit Sra and Reshad Hosseini. Conic geometric optimization on the manifold of positive definite matrices. *SIAM Journal on Optimization*, 25(1):713–739, 2015.

Suvrit Sra and Reshad Hosseini. Geometric optimization in machine learning. In *Algorithmic Advances in Riemannian Geometry and Applications*, pages 73–91. Springer, 2016.

Philip Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.