Riemannian Dictionary Learning and Sparse Coding for Positive Definite Matrices

Anoop Cherian  Suvrit Sra

Abstract—Data encoded as symmetric positive definite (SPD) matrices frequently arise in many areas of computer vision and machine learning. While these matrices form an open subset of the Euclidean space of symmetric matrices, viewing them through the lens of non-Euclidean Riemannian geometry often turns out to be better suited in capturing several desirable data properties. However, formulating classical machine learning algorithms within such a geometry is often non-trivial and computationally expensive. Inspired by the great success of dictionary learning and sparse coding for vector-valued data, our goal in this paper is to represent data in the form of SPD matrices as sparse conic combinations of SPD atoms from a learned dictionary via a Riemannian geometric approach. To that end, we formulate a novel Riemannian optimization objective for dictionary learning and sparse coding in which the representation loss is characterized via the affine invariant Riemannian metric. We also present a computationally simple algorithm for optimizing our model. Experiments on several computer vision datasets demonstrate superior classification and retrieval performance using our approach when compared to sparse coding via alternative non-Riemannian formulations.

Index Terms—Dictionary learning, Sparse coding, Riemannian distance, Region covariances

I. INTRODUCTION

Symmetric positive definite (SPD) matrices play an important role as data descriptors in several computer vision applications, for example in the form of region covariances [1]. Notable examples where such descriptors are used include object recognition [2], human detection and tracking [3], visual surveillance [4], 3D object recognition [5], among others. Compared with popular vector space descriptors, such as bag-of-words, Fischer vectors, etc., the second-order structure offered by covariance matrices is particularly appealing. For instance, covariances conveniently fuse multiple features into a compact form independent of the number of data points. By choosing appropriate features, this fusion can be made invariant to image affine distortions [6], or robust to static image noise and illumination variations; generating these descriptors is easy, for instance using integral image transforms [3].

We focus on SPD matrices for dictionary learning with sparse coding (DLSC) – a powerful tool in computer vision and machine learning [7] that enables state-of-the-art results for a variety of applications [8, 9].

Given a training set, the usual Euclidean dictionary learning problem seeks an overcomplete set of basis vectors (the dictionary) that can be linearly combined to sparsely represent each input data point; finding the sparse representation given a dictionary is termed sparse coding. While sparse coding was originally conceived for vector data from Euclidean spaces, there have been recent extensions of the setup to other data geometries, such as histograms [10], Grassmannians [11], and SPD matrices [12, 13, 14, 15]. The focus of this paper is on dictionary learning and sparse coding of SPD matrix data using a novel mathematical model inspired by the geometry of SPD matrices.

SPD matrices are an open subset of the Euclidean space of symmetric matrices. This may lead one to believe that a direct extension of usual sparse coding setup may suffice. However, there are several specific properties that applications using SPD matrices demand. For example, in DT-MRI applications, the semi-definite matrices are required to be at infinite distances from SPD matrices [16]. Such properties can be achieved by assuming a non-trivial geometry on these matrices via defining a suitable similarity measure; a few commonly used measures are (i) the affine invariant Riemannian metric [17], (ii) the log-Euclidean Riemannian metric [18], and (iii) the Stein metric [13], and (iv) the Jeffrey’s divergence [19]. Of these measures, the affine invariant Riemannian metric is the only intrinsic Riemannian metric that corresponds to a geodesic distance.
on the manifold of SPD matrices. While the log-Euclidean metric is also a Riemannian metric, it is extrinsic to the manifold in that the distances are computed as geodesics in the ambient Euclidean space (chordal distance), and can be shown to be a lower-bound to the intrinsic geodesic distance [20, Ch. 6]. The other measures described above are neither Riemannian and nor are their distances geodesic; as a result, their geometries are less well-understood.

We study dictionary learning and sparse coding of SPD matrices in their most natural geometry, via the affine invariant Riemannian metric. However, compared to the Euclidean setup, the geometry defined by this metric poses some unique challenges: (i) the manifold defined by this metric is negatively curved [21] and thus the geodesics are no more straight-lines; and (ii) in contrast to the Euclidean DLSC formulations, the objective function aligned with this metric is not convex in either the sparse coding phase or in the dictionary learning phase. We will present several theoretical properties of our formulation and demonstrate theoretically some situations where the formulation is convex. Figure 1 conceptually characterizes our goal.

Let us briefly mention the key contributions of this paper.

- **Formulation**: We propose a new model to learn a dictionary of SPD atoms; each data point is represented as a nonnegative sparse linear combination of SPD atoms from this dictionary. The representation quality is measured by the squared intrinsic Riemannian distance. As a theoretical refinement to our model, we mention an intuitive geometric constraint under which the nonconvex Riemannian sparse coding task actually becomes convex.

- **Optimization**: The main challenge in using our formulation is its higher computational cost relative to Euclidean sparse coding. However, we describe a simple and effective approach for optimizing our objective function. Specifically, we propose a dictionary learning algorithm on SPD atoms via conjugate gradient descent on the product manifold generated by the SPD atoms in the dictionary.

- **Experiments**: We present results on a few computer vision tasks on several state-of-the-art datasets to demonstrate superior performance obtained by using our new dictionary learning and sparse coding model. A forerunner to this paper appeared in [22]. The current paper differs from our conference paper in the following main aspects: (i) we propose a novel dictionary learning formulation and an efficient solver for it; and (ii) extensive experiments using our dictionary learning is also included in the experiments, and the entire experimental section reevaluated under the new setup, also including other datasets and evaluation metrics.

To set the stage for presenting our contributions, we first survey some recent methods suggested for sparse coding. After that we review key tools from Riemannian geometry that we will use to develop our ideas. Throughout we work with real matrices; extension to Hermitian positive definite matrices is straightforward. The space of $d \times d$ SPD matrices is denoted as $\mathcal{S}_+^d$, symmetric matrices by $\mathcal{S}^d$, and the space of (real) invertible matrices by $\text{GL}(d)$. By $\text{Log}(X)$, for $X \in \mathcal{S}_+^d$, we mean the principal matrix logarithm.

## II. Related Work

Dictionary learning and sparse coding (DLSC) of SPD matrices has received significant attention in the vision community due to the performance gains it brings to the respective applications [7, 12, 15]. Given a training dataset $\mathcal{X}$, the DLSC problem seeks a dictionary $\mathcal{B}$ of basis atoms, such that each data point $x \in \mathcal{X}$ can be approximated by a sparse linear combination of these atoms while minimizing a suitable loss function. Formally, the DLSC problem can be abstractly written as

$$
\min_{B, \theta_x \forall x \in \mathcal{X}} \sum_{x \in \mathcal{X}} \mathcal{L}(x, B, \theta_x) + \lambda \text{Sp}(\theta_x),
$$

where the loss function $\mathcal{L}$ measures the approximation quality obtained by using the "code" $\theta_x$, while $\lambda$ regulates the impact of the sparsity penalty $\text{Sp}(\theta_x)$.

As alluded to earlier, the manifold geometry hinders a straightforward extension of classical DLSC techniques (such as [23, 24]) to data points drawn from a manifold. Prior methods typically use surrogate similarity metrics that bypass the need to operate within the intrinsic Riemannian geometry, e.g., (i) by adapting information geometric divergence measures such as the log-determinant divergence or the Stein metric, (ii) by using extrinsic metrics such as the log-Euclidean metric, and (iii) by relying on the kernel trick to embed the SPD matrices into a suitable RKHS. We briefly review each of these schemes below.

### Statistical measures

In [14] and [25], a DLSC framework is proposed based on the log-determinant divergence (Burg loss) to model the loss. Since this divergence acts as a base measure for the Wishart distribution [4]—a natural probability density on SPD matrices—a loss defined using it is statistically well-motivated. The sparse coding formulation using this loss reduces to a MAXDET optimization problem [14] and is solved using interior-point methods. Unsurprisingly, the method is often seen to be computationally demanding even for moderately sized covariances (more than $5 \times 5$). Ignoring the specific manifold geometry of SPD matrices, one may directly extend the Euclidean DLSC schemes to the SPD setting. However, a naïve use of Euclidean distance on SPD matrices is usually found inferior in performance. It is argued in [15] that approximating an SPD matrix as sparse conic combinations of positive semi-definite rank-one outer-products of the Euclidean dictionary matrix atoms leads to improved performance under the Euclidean loss. However, the resulting dictionary learning subproblem is nonconvex and the reconstruction quality is still measured using a Euclidean loss. Further, discarding the manifold geometry is often seen to showcase inferior results compared to competitive methods [22].

### Differential geometric schemes

Among the computationally efficient variants of Riemannian metrics, one of the most popular is the log-Euclidean metric $d_{\text{LE}}$ [16] defined for $X,Y \in \mathcal{S}_+^d$ as $d_{\text{LE}}(X,Y) := \|\text{Log}(X) - \text{Log}(Y)\|_F$. The Log operator maps an SPD matrix isomorphically and
diffeomorphically into the flat space of symmetric matrices; the distances in this space are Euclidean. DLSC with the squared log-Euclidean metric has been proposed in the past \[20\] with promising results. A similar framework was suggested recently \[27\] in which a local coordinate system is defined on the tangent space at the given data matrix. While, their formulation uses additional constraints that make their framework coordinate independent, their scheme restricts sparse coding to specific problem settings, such as an affine coordinate system.

**Kernelized Schemes:** In \[12\], a kernelized DLSC framework is presented for SPD matrices using the Stein divergence \[18\] for generating the underlying kernel function. As the Stein metric is a statistically well-motivated similarity distance with strong connections \[28\], \[18\] to the natural Riemannian metric while being computationally superior, performances using this measure are expected to be similar to those using the Riemannian metric \[29\]. However, this measure does not produce geodesically exponential kernels for all bandwidths \[18\] making it less theoretically appealing. In \[2\], \[13\] kernels based on the log-Euclidean metric are proposed. A general DLSC setup is introduced for the more general class of Riemannian manifolds in \[30\]. The main goal of all these approaches is to linearize the curved manifold geometry by projecting the SPD matrices into an infinite dimensional Hilbert space as defined by the respective kernel. However, as recently shown theoretically in \[2\], \[31\] most of the curved Riemannian geometries (including the span of SPD matrices) do not have such kernel maps, unless the geometry is already isometric to the Euclidean space (as in the case of the log-Euclidean metric). This result severely restricts the applicability of traditional kernel methods to popular Riemannian geometries (which are usually curved), thereby providing strong motivation to study the standard machine learning algorithms within their intrinsic geometry — as is done in the current paper.

In light of the above summary, our scheme directly uses the affine invariant Riemannian metric to design our sparse reconstruction loss. To circumvent the computational difficulty we propose an efficient algorithm based on spectral projected gradients for sparse coding, while we use an adaptation of the non-linear conjugate gradient on manifolds for dictionary learning. Our experiments demonstrate that our scheme is computationally efficient and provides state of the art results on several computer vision problems that use covariance matrices.

III. **PRELIMINARIES**

We provide below a brief overview of the Riemannian geometry of SPD matrices. A manifold \(\mathcal{M}\) is a set of points endowed with a locally-Euclidean topology. A tangent vector at \(P \in \mathcal{M}\) is defined as the tangent to some curve in the manifold passing through \(P\). A tangent space \(T_P\mathcal{M}\) defines the union of all such tangent vectors to all possible such curves passing through \(P\); the point \(P\) is termed the *foot* of this tangent space. The dimensionality of \(T_P\mathcal{M}\) is the same as that of the manifold. It can be shown that the tangent space is isomorphic to the Euclidean space \[32\]; thus, they provide a locally-linear approximation to the manifold at its foot.

A manifold becomes a Riemannian manifold if its tangent spaces are endowed with a smoothly varying inner product. The Euclidean space, endowed with the classical inner product defined by the trace function (i.e., for two points \(X, Y \in \mathbb{S}^d\), \((X, Y) = \text{Tr}(XY)\)), is a Riemannian manifold. Recall that an SPD matrix has the property that all its eigenvalues are real and positive, and it belongs to the interior of a convex self-dual cone. Since for \(d \times d\) SPD matrices, this cone is a subset of the \(\frac{1}{2}(d(d+1))\)-dimensional Euclidean space of symmetric matrices, the set of SPD matrices naturally forms a Riemannian manifold under the trace metric. However, under this metric, the SPD manifold is not complete \[1\]. This is because, the trace metric does not enclose all Cauchy sequences originating from the interior of the SPD cone \[33\].

A possible remedy is to change the geometry of the manifold such that positive semi-definite matrices (which form the closure of SPD matrices for Cauchy sequences) are at an infinite distance to points in the interior of the SPD cone. This can be achieved by resorting to the classical *log-barrier function* \(g(P) = -\log \text{det}(P)\), popular in the optimization community in the context of interior point methods \[34\]. The trace metric can be modified to the new geometry induced by the barrier function by incorporating the curvature through its Hessian operator \(H_P = g''(P) = P^{-1}dP P^{-1}dP\). The Riemannian metric at \(P\) for two points \(Z_1, Z_2 \in T_P\mathcal{M}\) is therefore

\[
\langle Z_1, Z_2 \rangle_P = \langle H_P Z_1, Z_2 \rangle = \text{Tr}(P^{-1} Z_1 P^{-1} Z_2). \tag{2}
\]

There are two fundamental operations that one needs for computations on Riemannian manifolds: (i) the exponential map \(\text{Exp}_P : \mathbb{S}^d \rightarrow \mathbb{S}^d_+\); and (ii) the logarithmic map \(\text{Log}_P := \text{Exp}_P^{-1} : \mathbb{S}^d_+ \rightarrow \mathbb{S}^d\), where \(P \in \mathbb{S}^d_+\). The former projects a symmetric point on the tangent space onto the manifold (termed a *retraction*), the latter does the reverse. Note that these maps depend on the point \(P\) at which the tangent spaces are computed. In our analysis, we will be measuring distances assuming \(P\) to be the identity matrix, \(I\), in which case we will omit the subscript.

Note that the Riemannian metric provides a measure for computing distances on the manifold. Given two points on the manifold, there are infinitely many paths connecting them, of which the shortest path is termed the *geodesic*. It can be shown that the SPD manifold under the Riemannian metric in \[9\] is non-positively curved (Hadamard manifold) and has a unique geodesic between every distinct pair of points \[35\] (Chap. 12), \[20\] (Chap. 6). For \(X, Y \in \mathbb{S}^d_+\), there exists a closed form for this *geodesic distance*, given by

\[
d_R(X, Y) = \left\|\log(X^{-1/2} Y X^{-1/2})\right\|_F, \tag{3}
\]

where \(\log\) is the matrix logarithm. In the sequel, we will use \[3\] to measure the distance between input SPD matrices.

\[1\] A space is a complete metric space if all Cauchy sequences are convergent within that space.
matrices and their sparse coded representations obtained by combining dictionary atoms.

IV. PROBLEM FORMULATION

Let \( \mathbf{X} = \{ \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N \} \) denote a set of \( N \) SPD data matrices, where each \( \mathbf{X}_i \in \mathbb{S}^d_{+} \). Let \( \mathcal{M}_n^d \) denote the product manifold obtained from the Cartesian product of \( n \) SPD manifolds, i.e., \( \mathcal{M}_n^d = \mathbb{S}^d_{+} \times \cdots \times \mathbb{S}^d_{+} \subseteq \mathbb{R}^{d \times d \times n} \). Our goals are (i) to learn a third-order tensor (dictionary) \( \mathbf{B} \in \mathcal{M}_n^d \) in which each slice represents an SPD dictionary atom \( \mathbf{B}_j \in \mathbb{S}^d_{+} \), \( j = 1, 2, \ldots, n \); and (ii) to approximate each \( \mathbf{X}_i \) as a sparse conic combination of atoms in \( \mathbf{B} \); i.e., \( \mathbf{X}_i \sim \sum_{k} \mathbf{b}_k \alpha_{j_k} \) where \( \alpha_{j_k} \in \mathbb{R}^n_+ \) and \( \mathbf{b}_k \) for an \( n \)-dimensional vector \( v \). With this notation, our joint DLSC objective is

\[
\min_{\mathbf{B} \in \mathcal{M}_n^d} \sum_{k} \sum_{j=1}^{N} \frac{1}{2} d_R^2(\mathbf{X}_j, \mathbf{B}\alpha_{j}) + \text{Sp}(\mathbf{B}) + \Omega(\mathbf{B}),
\]

where \( \text{Sp} \) and \( \Omega \) are regularizers on the coefficient vectors \( \alpha_{j} \) and the dictionary tensor respectively.

Although formulation (4) may look complicated, it is a direct analogue of the vectorial DLSC setup to matrix data. For example, instead of learning a dictionary matrix in the vectorial DLSC, we learn a third-order tensor dictionary since our data \( \mathbf{X} \) are now matrices. The need to constrain the sparse coefficients to the non-negative orthant is required to make sure the linear combination of SPD atoms stays within the SPD cone. However, in contrast to the vectorial DLSC formulations for which the subproblems on the dictionary learning and sparse coding are convex separately, the problem in (4) is neither convex in itself, nor are its subproblems convex.

From a practical point of view, this lack of convexity is not a significant concern as all we need is a set of dictionary atoms which can sparse code the input. To this end, we propose below an alternating minimization (descent) scheme that alternates between locally solving the dictionary learning and sparse coding sub-problems, while keeping fixed the variables associated with the other.

A. Dictionary Learning Subproblem

Assuming the coefficient vectors \( \alpha \) available for all the data matrices, the subproblem for updating the dictionary atoms can be separated from (4) and written as:

\[
\min_{\mathbf{B} \in \mathcal{M}_n^d} \Theta(\mathbf{B}) := \frac{1}{2} \sum_{j=1}^{N} d_R^2(\mathbf{X}_j, \mathbf{B}\alpha_{j}) + \Omega(\mathbf{B}),
\]

\[
= \frac{1}{2} \sum_{j=1}^{N} \left\| \log(\mathbf{X}_j^{-\frac{1}{2}} \mathbf{B}\alpha_{j} \mathbf{X}_j^{-\frac{1}{2}}) \right\|_F^2 + \Omega(\mathbf{B}).
\]

1) Regularizers: Before delving into algorithms for optimizing (6), let us recall a few potential regularizers \( \Omega \) on the dictionary atoms, which are essential to avoid overfitting the dictionary to the data. For SPD matrices, we have several regularizers available, such as: (i) the largest eigenvalue regularizer \( \Omega(\mathbf{B}) = \sum_i \| \mathbf{B}_i \|_2^2 \), (ii) deviation of the dictionary from the identity matrix \( \Omega(\mathbf{B}) = \sum_i \| \mathbf{B}_i - I \|_F^2 \), (iii) the Riemannian elasticity regularizer \( \Omega(\mathbf{B}) = \sum_i \| \log(\mathbf{B}_i) - \log(I) \|_F^2 = d_R(\mathbf{B}_i, I)^2 \), and (iv) the trace regularizer, i.e., \( \Omega(\mathbf{B}) = \lambda \text{Tr}(\mathbf{B}) \), for a regularization parameter \( \lambda \). In the sequel, we use the unit-trace regularizer as it is simpler and performs well empirically.

2) Optimizing Dictionary Atoms: Among several first-order alternatives for optimizing over the SPD atoms (such as the steepest-descent, trust-region methods [37], etc.), the Riemannian Conjugate Gradient (CG) method [32][Chap.8], was found to be empirically more stable and faster. Below, we provide a short exposition of the CG method in the context of minimizing over \( \mathbf{B} \) which belongs to an SPD product manifold.

For an arbitrary non-linear function \( \theta(x), x \in \mathbb{R}^n \), the CG method uses the following recurrence at step \( k + 1 \)

\[
x_{k+1} = x_k + \gamma_k \xi_k,
\]

where the direction of descent \( \xi_k \) is

\[
\xi_k = - \nabla \theta(x_k) + \mu_k \xi_{k-1},
\]

with \( \nabla \theta(x_k) \) defining gradient of \( \theta \) at \( x_k \) \( \xi_0 = - \nabla \theta(x_0) \), and \( \mu_k \) given by

\[
\mu_k = \frac{(\nabla \theta(x_k))^T(\nabla \theta(x_k) - \nabla \theta(x_{k-1}))}{\nabla \theta(x_{k-1})^T \nabla \theta(x_{k-1})}.
\]

The step-size \( \gamma_k \) in (7) is usually found via an efficient line-search method [38]. It can be shown that [38][Sec.1.6] when \( \theta \) is quadratic with a Hessian \( Q \), the directions generated by (9) will be Q-conjugate to previous directions of descent \( \xi_0, \xi_1, \ldots, \xi_{k-1} \) thereby (7) providing the exact minimizer of \( f \) in fewer than \( d \) iterations (\( d \) is the manifold dimension).

For \( \mathbf{B} \in \mathcal{M}_n^d \) and referring back to (6), the recurrence in (7) will use the Riemannian retraction [32][Chap.4] and the gradient \( \nabla \theta(\mathbf{B}_k) \) will assume the Riemannian gradient (here we use \( \mathbf{B}_k \) to represent the dictionary tensor at the \( k \)-th iteration). This leads to an important issue: the gradients \( \nabla \theta(\mathbf{B}_k) \) and \( \nabla \theta(\mathbf{B}_{k-1}) \) belong to two different tangent spaces \( T_{\mathbf{B}_k} \mathcal{M} \) and \( T_{\mathbf{B}_{k-1}} \mathcal{M} \) respectively, and thus cannot be combined as in (9). Thus, following [32][Chapter 8] we resort to vector transport – a scheme to transport a tangent vector at \( P \in \mathcal{M} \) to a point \( \text{Exp}_P(S) \) where \( S \in T_P \mathcal{M} \) and \( \text{Exp} \) is the exponential map. The resulting formula for the direction update becomes

\[
\xi_{B_k} = - \nabla \theta(\mathbf{B}_k) + \mu_k \xi_{\gamma_k \xi_{k-1}}(\xi_{k-1}),
\]

where

\[
\mu_k = \frac{\langle \nabla \theta(\mathbf{B}_k), \nabla \theta(\mathbf{B}_k) - \xi_{\gamma_k \xi_{k-1}}(\nabla \theta(\mathbf{B}_{k-1})) \rangle}{\langle \nabla \theta(\mathbf{B}_{k-1}), \nabla \theta(\mathbf{B}_{k-1}) \rangle}.
\]

Here for \( Z_1, Z_2 \in T_P \mathcal{M} \), the map \( \xi_{Z_1}(Z_2) \) defines the vector transport given by:

\[
\xi_{Z_1}(Z_2) = \frac{d}{dt} \exp_P(Z_1 + tZ_2) \bigg|_{t=0}.
\]

The remaining technical detail is the expression for the Riemannian gradient \( \nabla \theta(\mathbf{B}) \), which we derive next.
3) Riemannian Gradient: The following lemma connects the Riemannian gradient to the Euclidean gradient of \( \Theta(B) \) in (3).

**Lemma 1.** For a dictionary tensor \( B \in M^d \), let \( \Theta(B) \) be a differentiable function. Then the Riemannian gradient \( \nabla \Theta(B) \) satisfies the following condition:

\[
\langle \nabla \Theta(B), \zeta \rangle_B = \langle \nabla \Theta(B), \zeta \rangle, \quad \forall \zeta \in T_p M^d,
\]

where \( \nabla \Theta(B) \) is the Euclidean gradient of \( \Theta(B) \). The Riemannian gradient for the \( i \)-th dictionary atom is given by \( \nabla \Theta(B) = B_i \nabla B_i \Theta(B) B_i \).

**Proof:** See [32] (Chap. 5). The latter expression is obtained by substituting the inner product on the LHS of (13) by its definition in (3).

We can derive the Euclidean gradient \( \nabla \Theta(B) \) as follows: let \( S_j = X_j^{-\frac{1}{2}} \) and \( M_j(B) = B \alpha_j = \sum_{i=1}^n \alpha_{j,i} B_i \). Then,

\[
\Theta(B) = \frac{1}{2} \sum_{j=1}^N \log(S_j M_j(B) S_j) + \lambda_B \sum_{i=1}^n \log(B_i).
\]

The derivative \( \nabla B_i \Theta(B) \) of (14) w.r.t. to atom \( B_i \) is:

\[
\sum_{j=1}^N \alpha_{j,i} \left(S_j \log(M_j(B)) (M_j(B))^{-1} S_j\right) + \lambda_B I.
\]

**B. Sparse Coding Subproblem**

Referring back to (4), let us now consider the sparse coding subproblem. Suppose we have a dictionary tensor \( B \) available. For a data matrix \( X_j \in S^d \) our sparse coding objective is to solve

\[
\min_{\alpha_j \geq 0} \phi(\alpha_j) := \frac{1}{2} d_K^2(X_i, B \alpha_j) + \text{Sp}(\alpha_j)
\]

\[
= \frac{1}{2} \left\| \log \sum_{i=1}^n \alpha_{j,i} X_i^{-\frac{1}{2}} B_j X_i^{-\frac{1}{2}} \right\|_F^2 + \text{Sp}(\alpha_j),
\]

where \( \alpha_{j,i} \) is the \( i \)-th dimension of \( \alpha_j \) and \( \text{Sp} \) is a sparsity inducing function. For simplicity, we use the sparsity penalty \( \text{Sp}(\alpha) = \lambda \|\alpha\|_1 \), where \( \lambda > 0 \) is a regularization parameter. Since we are working with \( \alpha \geq 0 \), we replace this penalty by \( \lambda \sum_i \alpha_i \), which is differentiable.

The subproblem (16) measures reconstruction quality offered by a sparse non-negative linear combination of the atoms to a given input point \( X \). It will turn out (see experiments in Section [5]) that the reconstructions obtained via this model actually lead to significant improvements in performance over sparse coding models that ignore the rich geometry of SPD matrices. But this gain comes at a price: model (16) is a nontrivial to optimize; it remains difficult even if we take into account geodesic convexity of \( d_K \).

While in practice this nonconvexity does not seem to hurt our model, we show below a surprising but intuitive constraint under which Problem (16) actually becomes convex. The following lemma will be useful later.

**Lemma 2.** Let \( B, C, \text{ and } X \) be fixed SPD matrices. Consider the function \( f(x) = d_K^2(xB + C, X) \). The derivative \( f'(x) \) is given by

\[
f'(x) = 2 \text{Tr}(\log(S(xB + C)S^{-1}(xB + C)^{-1}BS)),
\]

where \( S = X^{-\frac{1}{2}} \).

**Proof:** Introduce the shorthand \( M(x) \equiv xB + C \), from definition (19) and using \( \|Z\|^2_F = \text{Tr}(Z^T Z) \) we have

\[
f(x) = \text{Tr}(\log(SM(x)S^T S \log(SM(x)S))).
\]

The chain-rule of calculus then immediately yields

\[
f'(x) = 2 \text{Tr}(\log(SM(x)S)(SM(x)S)^{-1}SM'(x)S),
\]

which is nothing but (17).

As a brief digression, let us mention below an interesting property of the sparse-coding problem. We do not exploit this property in our experiments, but highlight it here for its theoretical appeal.

**Theorem 3.** The function \( \phi(\alpha) := d_K^2(\sum_i \alpha_i B_i, X) \) is convex on the set

\[
A := \{ \alpha \mid \sum_i \alpha_i B_i \preceq X, \text{ and } \alpha \geq 0 \}.
\]

**Proof:** See Appendix [A].

Let us intuitively describe what Theorem 3 is saying. While sparsely encoding data we are trying to find sparse coefficients \( \alpha_1, \ldots, \alpha_n \), such that in the ideal case we have \( \sum_i \alpha_i B_i = X \). But in general this equality cannot be satisfied, and one only has \( \sum_i \alpha_i B_i \approx X \), and the quality of this approximation is measured using \( \phi(\alpha) \) or some other desirable loss-function. The loss \( \phi(\alpha) \) from (16) is nonconvex while convexity is a “unilateral” property—it lives in the world of inequalities rather than equalities (39).

And it is known that SPD matrices in addition to forming a manifold also enjoy a rich conic geometry that is endowed with the Löwner partial order. Thus, instead of seeking arbitrary approximations \( \sum_i \alpha_i B_i \approx X \), if we limit our attention to those that underestimate \( X \) as in (18), we might benefit from the conic partial order. It is this intuition that Theorem 3 makes precise.

1) Optimizing Sparse Codes: Writing \( M(\alpha_p) = \alpha_p B_p + \sum_{i \not= p} \alpha_i B_i \) and using Lemma 2 we obtain

\[
\frac{\partial \phi(\alpha)}{\partial \alpha_p} = \text{Tr} \left( \log(SM(\alpha_p)S)(SM(\alpha_p)S)^{-1}SB_pS \right) + \lambda.
\]

Computing (19) for all \( \alpha_p \) in conjunction with a gradient-projection scheme essentially runs the iteration

\[
\alpha^{k+1} \leftarrow \mathcal{P}[\alpha^k - \eta_k \nabla \phi(\alpha^k)], \quad k = 0, 1, \ldots
\]
Algorithm 1: Efficient computation of gradients

\begin{verbatim}
Input: \( B_1, \ldots, B_n, X \in S^d_+ \), \( \alpha \geq 0 \)
\( S \leftarrow X^{-1/2}; M \leftarrow \sum_{i=1}^{n} \alpha_i B_i; \)
\( T \leftarrow S \log((SM)(MS)^{-1}); \)
\( \text{for } i = 1 \text{ to } n \text{ do} \)
\( \quad g_i \leftarrow \text{Tr}(TB_p); \)
\( \text{end} \)
\( \text{return } g \)
\end{verbatim}

where \( \mathcal{P}[\cdot] \) denotes the projection operator defined as
\[ \mathcal{P}[\alpha] \equiv \alpha \rightarrow \arg\min_{\alpha'} \frac{1}{2} ||\alpha' - \alpha||^2_2, \quad \text{s.t. } \alpha' \geq 0, \alpha' \in \mathcal{A}. \] (21)

Iteration (20) has three major computational costs: (i) the stepsize \( \eta_k \); (ii) the gradient \( \nabla \phi(\alpha_k) \); and (iii) the projection (21). Alg. 1 shows how to efficiently obtain the gradient. The projection task (21) is a special least-squares (dual) semidefinite program (SDP), which can be solved using any SDP solver or by designing a specialized routine. To avoid the heavy computational burden imposed by an SDP, we drop the constraint \( \alpha \in \mathcal{A} \); this sacrifices convexity but makes the computation vastly easier, since with this change, we simply have \( \mathcal{P}[\alpha] = \max(0, \alpha) \).

In (20), it only remains to specify how to obtain the stepsize \( \eta_k \). There are several choices available in the nonlinear programming literature (38) for choosing \( \eta_k \), but most of them can be quite expensive. In our quest for an efficient sparse coding algorithm, we choose to avoid expensive line-search algorithms for selecting \( \eta_k \) and prefer to use the Barzilai-Borwein stepsizes (40), which can be computed in closed form and lead to remarkable gains in performance (40). In particular, we use the Spectral Projected Gradient (SPG) method (42) by adapting a simplified implementation of (41).

SPG runs iteration (20) using Barzilai-Borwein stepsizes with an occasional call to a nonmonotone line-search strategy to ensure convergence of \( \{\alpha^k\} \). Without the constraint \( \alpha' \in \mathcal{A} \), we cannot guarantee anything more than a stationary point of (4). While if we were to use the additional constraint then we can even obtain global optimality for iterates generated by (20).

V. Experiments

In this section, we provide experiments on simulated and real-world data demonstrating the effectiveness of our algorithm compared to the state-of-the-art DLSC methods on SPD valued data. First, we demonstrate results on simulated data analyzing the performance of our framework for various settings. This will precede experiments on standard benchmark datasets.

A. Comparison Methods

In the experiments to follow, we will denote dictionary learning and sparse coding algorithms by DL and SC respectively. We will compare our Riemannian (Riem) formulation against combinations of several state-of-the-art DLSC methods on SPD matrices, namely using (i) log-Euclidean (LE) metric for DLSC (26), (ii) Frobenius norm (Frob) which discards the manifold structure, (iii) kernel methods such as the Stein-Kernel (18) proposed in (12) and the log-Euclidean kernel (13). We refrain from comparisons to other earlier DLSC methods such as tensor sparse coding (TSC) (43) and generalized dictionary learning (GDL) (15) as their performance is generally seen to be inferior to the above methods (22).

B. Simulated Experiments

In this subsection, we evaluate in a controlled setting, some of the properties of our Riemannian sparse coding scheme. For all our simulations, we used covariances generated from data vectors sampled from a zero-mean unit covariance normal distribution. For each covariance sample, the number of data vectors is chosen to be ten times the dimensions typically use small SPD matrices (3 \( \times \) 3), the dimensionality is very diverse for other applications in computer vision. For example, Gabor covariances for face recognition uses about 40-dimensional SPD matrices (43), while even larger covariance descriptors are becoming common (44). The goal of this experiment is to analyze the scalability of our sparse coding setup against an increasing size of the data matrices. To this end, we fixed the number of dictionary atoms to be 200, while increased the matrix dimensionality from 3 to 100. Figure 2(a) plots the time-taken by our method against the na"ive Frob-SC method (although it uses the SPG method for solution). The plot shows that the extra computations required by our Riem-SC is not substantial compared to Frob-SC.

1) Increasing Data Dimensionality: While DT-MRI applications typically use small SPD matrices (3 \( \times \) 3), the dimensionality is very diverse for other applications in computer vision. For example, Gabor covariances for face recognition uses about 40-dimensional SPD matrices (43), while even larger covariance descriptors are becoming common (44). The goal of this experiment is to analyze the scalability of our sparse coding setup against an increasing size of the data matrices. To this end, we fixed the number of dictionary atoms to be 200, while increased the matrix dimensionality from 3 to 100. Figure 2(a) plots the time-taken by our method against the na"ive Frob-SC method (although it uses the SPG method for solution). The plot shows that the extra computations required by our Riem-SC is not substantial compared to Frob-SC.

2) Increasing Dictionary Size: In this experiment, we compare the scalability of our method to work with larger dictionary tensors. To this end, we fixed the data dimensionality to 10, while increased the number of dictionary atoms from 20 to 1000. Figure 2(b) plots the time-taken...
against the dictionary size. As is expected, the sparse coding performance for all the kernelized schemes drops significantly for larger dictionary sizes, while our scheme performs fairly.

3) Increasing Sparsity Regularization: In this experiment, we decided to evaluate the effect of the sparsity promoting regularization \( \lambda \) in (16). To this end, we generated a dictionary of 100 atoms from covariances of Gaussian random variables. Later, 1000 SPD matrices are produced using conic combinations of randomly selected atoms. We used an active size of 10 dictionary atoms for all the SPD matrices. After adding random SPD noise to each matrix, we used half of them for learning the dictionary, while the other half was used for evaluating the sparsity regularization. We increased \( \lambda \) from \( 10^{-5} \) to \( 10^{5} \) at steps of 10. In Figure 5(a), we plot the sparsity (i.e., number of non-zero coefficients/size of coefficients) for varying \( \lambda \). We see that while the lower values of \( \lambda \) do not have much influence on sparsity, as \( \lambda \) increases beyond a certain threshold, sparsity increases. A similar trend is seen for increasing data dimensionality. However, we find that the influence of \( \lambda \) starts diminishing as the dimensionality increases. For example, sparsity plateaus after 3% for 5-dimensional data, while this happens at nearly 15% for 20-dimensional data. The plateauing of sparsity is not unexpected and is directly related to the Riemannian metric that we use – our loss will prevent all the sparse coefficients from going to zero simultaneously as in such a case the objective will tend to infinity. Further, as the matrix dimensionality increases, it is more likely that the data matrices become ill-conditioned. As a result, this plateau-ing happens much earlier than for better conditioned matrices (as in the case of 5-dimensional matrices in Figure 5(a)).

In Figure 5(b) we contrast the sparsity pattern produced by our Riemannian sparse coding (Riem-DL + Riem-SC) scheme against that of the traditional sparse coding objective using log-Euclidean sparse coding (LE-DL + LE-SC), for 20-dimensional SPD data. As is expected, the log-Euclidean DL follows the conventional convergence patterns in which sparsity goes to zero for larger values of the regularization. Since for larger regularizations, most of the coefficients in our Riem-SC have low values, we can easily discard them by thresholding. However, we believe this difference in the sparsity patterns needs to be accounted for when choosing the regularization parameters for promoting sparsity in our setup.

4) Convergence for Increasing Dimensionality: In this experiment, we evaluate the convergence properties of our dictionary learning sub-problem based on the Riemannian conjugate gradient scheme. To this end, we used the same setup as in the last experiment using data generated by a pre-defined dictionary, but of different dimensionality \( \in \{3, 5, 10, 20\} \). In Figure 5 we plot the dictionary learning objective against the iterations. As is expected, smaller data dimensionality shows faster convergence. That said, even 20-dimensional data was found to converge in less than 50 alternating iterations of the algorithm, which is remarkable.

C. Experiments with Public Datasets

Now let us evaluate the performance of our framework on computer vision datasets. We experimented on data available from four standard computer vision applications, namely (i) 3D object recognition on the RGBD objects dataset [45], (ii) texture recognition on the standard Brodatz dataset [46], (iii) person re-identification on the ETHZ people dataset [47], and (iv) face recognition on the Youtube faces dataset [48]. We describe these datasets below.

1) Brodatz Texture: Texture recognition is one of the most successful applications of covariance descriptors [49]. For this evaluation, we used the Brodatz texture dataset \( \{ \} \), from which we took 100 gray scale texture images, each of dimension 512 \( \times \) 512. We extracted 32 \( \times \) 32 patches from a dense grid without overlap thus generating 256 texture patches per image, and totalling 25600 patches in our dataset. To generate covariance descriptors from each patch, we followed the traditional protocol, i.e., we extracted a 5-dimensional feature descriptor from each pixel location in each patch. The features are given by: \( F_{\text{texture}} = [x, y, I, \text{abs}(I_x), \text{abs}(I_y)]^T \), where the first two dimensions are the coordinates of a pixel from the top-left corner of a patch, the last three dimensions are the image intensity, and image gradients in the \( x \) and \( y \) directions respectively. Region covariances of size 5 \( \times \) 5 are computed from all features in a patch.

2) ETHZ Person Re-identification Dataset: Tracking and identifying people in severely dynamic environments from multiple cameras play an important role in visual surveillance. The visual appearances of people in such applications are often noisy, and low resolution. Further, the appearances undergo drastic variations with respect to their pose, scene illumination, and occlusions. Lately, covariance descriptors have been found to provide a robust setup for this task [12]. In this experiment, we evaluate the performance of clustering people appearances on the benchmark ETHZ dataset [51]. This dataset consists of low-resolution images of tracked people from a real-world surveillance setup. The images are from 146 different individuals. There are about 5–356 images per person. Sample images from this dataset are shown in Figure 4. There are a total of 8580 images in this dataset.

Our goal in this experiment is to evaluate the performance of our DLSC framework to learn generic dictionaries on covariance descriptors produced from this application. Note that some of the classes in this dataset does not have enough instances to learn a specific dictionary for them. Several types of features have been suggested in literature for generating covariances on this dataset that have shown varying degrees of success such as Gabor wavelet based features [50], color gradient based features [12], etc. Rather than detailing the results on several feature combinations, we describe here the feature combination that worked the best in our experiments. For this purpose, we used a validation set of 500 covariances and 10 true clusters from this dataset. The performance was evaluated using

http://www.ux.uis.no/~tranzen/brodatz.html
Fig. 2. Sparse coding time against (a) increasing matrix dimensionality and (b) increasing number of dictionary atoms and. We used a maximum of 100 iterations for all the algorithms.

Fig. 3. Sparsity of coding against (a) increasing sparsity inducing regularization $\lambda$ for various matrix dimensionality and (b) sparsity against lambda in comparison to that for log-Euclidean DL.

Fig. 5. Dictionary Learning objective using Riemannian conjugate gradient descent against increasing number of iterations (alternating with the sparse coding sub-problem). We plot the convergence of the objective for various dimensionality of the data matrices.

the Log-Euclidean SC setup with a dictionary learning via Log-Euclidean K-Means. We used a combination of nine features for each image as described below:

$$F_{ETHZ} = [x \ I_r \ I_g \ I_b \ Y_i \ \text{abs}(I_x) \ \text{abs}(I_y) \ \text{abs}(\sin(\theta) + \cos(\theta)) \ \text{abs}(H_y)] ,$$

where $x$ is the x-coordinate of a pixel location, $I_r, I_g, I_b$ are the RGB color of a pixel, $Y_i$ is the pixel intensity in the YCbCr color space, $I_x, I_y$ are the gray scale pixel gradients, and $H_y$ is the y-gradient of pixel hue. Further, we also use the gradient angle $\theta = \tan^{-1}(I_y/I_x)$ in our feature set. Each image is resized to a fixed size $300 \times 100$, and is divided into upper and lower parts. We compute two different region covariances for each part, which are combined as two block diagonal matrices to form a single covariance descriptor of size $18 \times 18$ for each appearance image.

3) 3D Object Recognition Dataset: The goal of this experiment is to recognize objects in 3D point clouds. To this end, we used the public RGB-D Object dataset [45], which consists of about 300 objects belonging to 51 categories and spread in about 250K frames. We used approximately 15K frames for our evaluation with approximately 250-350 frames devoted to every object seen from three different
view points (30, 45, and 60 degrees above the horizon). Following the procedure suggested in [52][Chap. 5], for every frame, the object was segmented out and 18-dimensional feature vectors generated for every 3D point in the cloud (and thus \(18 \times 18\) covariance descriptors); the features we used are as follows:

\[
F_{\text{RGBD}} = [x, y, z, I_r, I_g, I_b, I_{xx}, I_{yy}, I_{xy},
I_m, \delta_x, \delta_y, \delta_m, \nu_x, \nu_y, \nu_z, \nu_z],
\]

(23)

where the first three dimensions are the spatial coordinates, \(I_m\) is the magnitude of the intensity gradient, \(\delta\)’s represent gradients over the depth-maps, and \(\nu\) represents the surface normal at the given 3D point.

4) Youtube Faces Dataset: In this experiment, we evaluate the performance of the Riemannian DLSC setup to deal with a larger dataset of high-dimensional covariance descriptors for face recognition. To this end, we used the challenging Youtube faces dataset [48] that consists of 3425 short video clips of 1595 individuals, each clip containing between 48–6K frames. There are significant variations in head pose, context, etc. for each person across clips and our goal is to associate a face with its ground truth person label. We proceed by first cropping out face regions from the frames by applying a state-of-the-art face detector [53], which results in approximately 196K face instances. As most of the faces within a clip do not have significant variations, we subsample this set randomly to generate our dataset of \(\sim 43\)K face patches. Next, we convolved the image with a filter bank of 40 Gabor filters with 5 scales and 8 different orientations to extract the facial features for each pixel, generating \(40 \times 40\) covariances.

D. Experimental Setup

1) Evaluation Techniques: We evaluated our algorithms from two perspectives, namely (i) nearest neighbor (NN) retrieval against a gallery set via computing the Euclidean distances between sparse codes, and (ii) one-against-all classification using a linear SVM trained over the sparse codes. Given that computing the geodesic distance between SPD matrices is expensive, while the Frobenius distance between them results in poor accuracy, the goal of the first experiment is to evaluate the quality of sparse coding to approximate the input data in terms of codes that belong to the non-negative orthant of the Euclidean space – superior performance implying that the sparse codes provide efficient representations that could bypass the Riemannian geometry, and can enable other faster indexing schemes such as locality sensitive hashing for faster retrieval. Our second experiment evaluates the linearity of the space of sparse codes – note that they are much higher dimensional than the original covariances themselves and thus we expect them to be linearly separable in the sparse space.

2) Evaluation Metric: For classification experiments, we use the one-against-all classification accuracy as the evaluation metric. For NN retrieval experiments, we use the Recall@K accuracy, which is defined as follows. Given a gallery \(X\) and a query set \(Q\). Recall@K computes the average accuracy when retrieving \(K\) nearest neighbors from \(X\) for each instance in \(Q\). Suppose \(G^q_K\) stands for the set of ground truth class labels associated with the \(q\)th query, and if \(S^q_K\) denotes the set of labels associated with the \(K\)
neighbors found by some algorithm for the \( q \) queries, then

\[
Recall@K = \frac{1}{|Q|} \sum_{q \in Q} \frac{|G_q^k \cap S_q^k|}{|G_q^k|}. \tag{24}
\]

3) Data Split: All the experiments used 5-fold cross-validation in which 80% of the datasets were used for training the dictionary, 10% for generating the gallery set or as training set for the linear SVM, and the rest as the test/query points. We evaluate three setups for generating the dictionaries, (i) using a proper dictionary learning strategy, and (ii) using clustering the training set via K-Means using the appropriate distance metric, and (iii) random sampling of the training set.

4) Hyperparameters: The size of the dictionary was considered to be twice the number of classes in the respective dataset. This scheme was considered for all the comparison methods as well. We experimented with larger sizes, but found that performance generally almost saturates. This is perhaps because the datasets that we use already have a large number of classes, and thus the dictionary sizes generated using this heuristic makes them already significantly overcomplete. The other hyperparameter in our setup is the sparsity of the generated codes. As the different sparse coding methods (including ours and the methods that we compare to) have varied sensitivity to the regularization parameter, comparing all the methods to different sparsities turned out to be cumbersome. Thus, we decided to fix the sparsity of all methods to 10%-sparse and adjusted the regularization parameter for each method appropriately (on a small validation set separate from the training set).

5) Implementation Details: Our DLSC scheme was implemented in MATLAB. We used the ManOpt Riemannian geometric optimization toolbox [54] for implementing the CG method in our DL sub-problem. As our problem is non-convex, we found that initializing the dictionary learning setup using K-Means clustering (using the Karcher mean algorithm [17]) demonstrate faster convergence.

E. Results

In this section, we compare the performance of our dictionary learning (Riem-DL) and sparse coding (Riem-SC) method against several prior DLSC schemes on the four datasets that we described above. Our choice of comparison methods include (i) Riemannian geometric methods such as log-Euclidean (LE-DL + LE-SC), (ii) Kernelized methods
using the Stein kernel (Kernel-Stein-DL and kernel-Stein-SC), (iii) Euclidean DLSC (Frob-DL + Frob-SC), and using a dictionary generated by random sampling the dataset followed by sparse coding using our Riemannian method (Random-DL + Riem-SC). In Figure 9 we show the performance comparison for the task of K-NN where $K$ is increased from one to 25. In Figure 7 we show the performance for the one-against-all classification setup.

An alternative to dictionary learning that is commonly adopted is to approximate the dictionary by using the centroids of clusters generated from a K-Means clustering of the dataset. Such a method is faster in comparison to a Riemannian DL, while also demonstrate reasonable performance. Thus, an important experiment with regard to learning the dictionary is to make sure using dictionary learning provides superior performance compared to this ad hoc setup. In Figure 6 we plot the K-NN retrieval when we use a clustering scheme to generate the dictionary. In Figure 8 we show the same in a classification setup.

**F. Discussion of Results**

With regard to Figure 9 we found that the performance of different methods is diverse across datasets. For example, the log-euclidean DLSC variant (LE-DL+LE-SC) is generally seen to showcase good performance across datasets. However, its performance is inferior when the number of data instances per class is small (as in the ETHZ people dataset). The kernelized DLSC method (Kernel-Stein-DL) performed favorably on most datasets, except on the faces dataset which used larger covariances (which are close to being ill-conditioned). Perhaps, this performance drop is due to numerical instability issues that this method gets into when the large covariances are poorly conditioned. The most surprising of the results that we found was for the Frob-DL case. It is generally assumed that using Frobenius distance for comparing SPD matrices leads to poor accuracy, which we see in Figures 9(a) and 9(b). However, when the data dimensionality commensurates with the number of classes and when there is sufficient data to learn the dictionary (as in Figure 9(d)), (Frob-DL+Frob-SC) can lead to state-of-the-art performance. In comparison to all the above methods, Riem-DL+Riem-SC was found to produce consistent and competitive performance, substantiating the usefulness of our proposed method. While running the experiments, we found that the initialization of our DL sub-problem (from K-Means) played an important role in achieving this superior performance. Figure 7 demonstrates a similar pattern in the classification setting.

The usefulness of our Riem-DL is further evaluated against alternative DL schemes via clustering in Figure 6. We see that learning the dictionary using Riem-DL demonstrates the best performance against the next best and
efficient alternative of using the LE-KMeans that was done in [22]. Using Frob-KMeans or using a random dictionary are generally seen to have inferior performance compared to other learning methods. In Figure 8, a similar trend is seen in the classification setting.

VI. CONCLUSIONS

In this paper, we proposed a novel setup for dictionary learning and sparse coding of data in the form of SPD matrices. In contrast to prior methods that use proxy distance measures as similarity metrics to define the sparse coding approximation loss, our formulation used a loss driven by the natural Riemannian metric (affine invariant Riemannian metric) on the SPD manifold. We proposed an efficient adaptation of the well-known non-linear conjugate gradient method for learning the dictionary in the product space of SPD manifolds and a fast algorithm for sparse coding based on the spectral projected gradient. Our experiments on simulated and several benchmark computer vision datasets demonstrated the superior performance of our method against prior works; especially our results showed that learning the dictionary using our scheme leads to significantly better accuracy (in retrieval and classification) than other heuristic and approximate schemes to generate the dictionary.

APPENDIX

Here we prove Theorem 3.

Lemma 4. Let \( Z \in \text{GL}(d) \) and let \( X \in S^d_+ \). Then, \( Z^T X Z \in S^d_+ \).

Lemma 5. The Fréchet derivative [55, see e.g., Ch. 1] of the map \( X \mapsto \log X \) at a point \( Z \) in the direction \( E \) is given by

\[
D \log(Z)(E) = \int_0^1 (\beta Z + (1 - \beta)I)^{-1} E (\beta Z + (1 - \beta)I)^{-1} d\beta.
\] (25)

Proof: See e.g., [55, Ch. 11].

Corollary 6. Consider the map \( \ell(\alpha) := \alpha \in \mathbb{R}^n_+ \mapsto \text{Tr}(\log(SM(\alpha)HS)) \), where \( M \) is a map from \( \mathbb{R}^n_+ \to S^d_+ \) and \( H \in S^d \), \( S \in S^d_+ \). Then, for \( 1 \leq p \leq n \), we have

\[
\frac{\partial \ell(\alpha)}{\partial \alpha_p} = \int_0^1 \text{Tr}[K_\beta S \frac{\partial M(\alpha)}{\partial \alpha_p} S K_\beta H] d\beta,
\]

where \( K_\beta := (\beta SM(\alpha)S + (1 - \beta)I)^{-1} \).

Proof: Simply apply the chain-rule of calculus and use linearity of \( \text{Tr}(\cdot) \).
Lemma 7. The Fréchet derivative of the map $X \mapsto X^{-1}$ at a point $Z$ in direction $E$ is given by

$$D(Z^{-1})(E) = -Z^{-1}EZ^{-1}. \quad (26)$$

We are now ready to prove Theorem 3.

Thm. 3 We show that the Hessian $\nabla^2 \phi(\alpha) \succeq 0$ on $\mathcal{A}$. To ease presentation, we write $S = X^{-1/2}, M = M(\alpha) = \sum_i \alpha_i B_i$, and let $D_q$ denote the differential operator $D_{\alpha_q}$. Applying this operator to the first-derivative given by Lemma 2 (in Section V-B, we obtain (using the product rule) the sum

$$\text{Tr}([D_q \log(SMS)](SMS)^{-1} SB_p S) + \text{Tr}(\log(SMS)D_q[(SMS)^{-1} SB_p S]).$$

We now treat these two terms individually. To the first we apply Corr. 6. So

$$\text{Tr}([D_q \log(SMS)](SMS)^{-1} SB_p S) = \int_0^1 \text{Tr}(K_\beta SB_q SK_\beta(SMS)^{-1} SB_p S)d\beta = \int_0^1 \text{Tr}(SB_q SK_\beta(SMS)^{-1} SB_p SK_\beta)d\beta = \int_0^1 (\Psi_\beta(p), \Psi_\beta(q))_M d\beta,$$

where the inner-product $\langle \cdot, \cdot \rangle_M$ is weighted by $(SMS)^{-1}$ and the map $\Psi_\beta(p) := SB_p SK_\beta$. We find a similar inner-product representation for the second term too. Starting with Lemma 7 and simplifying, we obtain

$$\text{Tr}(\log(SMS)D_q[(SMS)^{-1} SB_p S]) = -\text{Tr}(\log(SMS)(SMS)^{-1} SB_q M^{-1} B_p S) = \text{Tr}(-S \log(SMS)S^{-1} M^{-1} B_q M^{-1} B_p) = \text{Tr}(M^{-1} B_p [-S \log(SMS)S^{-1}] M^{-1} B_q).$$

By assumption $\sum_i \alpha_i B_i = M \preceq X$, which implies $SMS \preceq I$. Since $\log(\cdot)$ is operator monotone [20], it follows that $\log(SMS) \preceq 0$; an application of Lemma 4 then yields $S \log(SMS)S^{-1} \preceq 0$. Thus, we obtain the weighted inner-product

$$\text{Tr}(M^{-1} B_p [-S \log(SMS)S^{-1}] M^{-1} B_q) = \langle M^{-1} B_p, M^{-1} B_q \rangle_L,$$

where $L = [-S \log(SMS)S^{-1}] \succeq 0$, whereby $\langle \cdot, \cdot \rangle_L$ is a valid inner-product.

Thus, the second partial derivatives of $\phi$ may be ultimately written as

$$\frac{\partial^2 \phi(\alpha)}{\partial \alpha_p \partial \alpha_q} = (\Gamma(B_q), \Gamma(B_p)),$$

for some map $\Gamma$ and some corresponding inner-product (the map and the inner-product are defined by our analysis above). Thus, we have established that the Hessian is a Gram matrix, which shows it is semidefinite. Moreover, if the $B_i$ are different ($1 \leq i \leq n$), then the Hessian is strictly positive definite.
[24] M. Elad and M. Aharon, “Image denoising via learned dictionaries and sparse representation,” in Computer Vision and Pattern Recognition, 2006.

[25] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, “Positive definite dictionary learning for region covariances,” in International Conference on Computer Vision. IEEE, 2011.

[26] K. Guo, P. Ishwar, and J. Konrad, “Action recognition using sparse representation on covariance manifolds of optical flow,” in International Conference on Advanced Video and Signal Based Surveillance. IEEE, 2010.

[27] J. Ho, Y. Xie, and B. Vemuri, “On a nonlinear generalization of sparse coding and dictionary learning,” in International Conference on Machine Learning, 2013.

[28] A. Cichocki, S. Cruces, and S.-i. Amari, “Log-determinant divergences revisited: Alpha-beta and gamma log-det divergences,” Entropy, vol. 17, no. 5, pp. 2988–3034, 2015.

[29] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos, “Jensen-bregman logdet divergence with application to efficient similarity search for covariance matrices,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 9, pp. 2161–2174, 2013.

[30] M. Harandi and M. Salzmann, “Riemannian coding and dictionary learning: Kernels to the rescue,” Computer Vision and Pattern Recognition, 2009.

[31] A. Feragen, F. Lauze, and S. Hauberg, “Geodesic exponential kernels: When curvature and linearity conflict,” arXiv preprint arXiv:1411.0296, 2014.

[32] P.-A. Absil, R. Mahony, and R. Sepulchre, Optimization algorithms on matrix manifolds. Princeton University Press, 2009.

[33] G. Cheng and B. C. Vemuri, “A novel dynamic system in the space of spd matrices with applications to appearance tracking,” SIAM journal on imaging sciences, vol. 6, no. 1, pp. 592–615, 2013.

[34] Y. E. Nesterov, M. J. Todd et al., “On the riemannian geometry defined by self-concordant barriers and interior-point methods,” Foundations of Computational Mathematics, vol. 2, no. 4, pp. 333–361, 2002.

[35] S. Lang, Fundamentals of differential geometry. Springer Science & Business Media, 2012, vol. 191.

[36] X. Pennec, R. Stefanescu, V. Arsigny, P. Fillard, and N. Ayache, “Riemannian elasticity: A statistical regularization framework for non-linear registration,” in International Conference on Medical Image Computing and Computer Assisted Interventions. Springer, 2005.

[37] P.-A. Absil, C. G. Baker, and K. A. Gallivan, “Trust-region methods on riemannian manifolds,” Foundations of Computational Mathematics, vol. 7, no. 3, pp. 303–330, 2007.

[38] D. P. Bertsekas and D. P. Bertsekas, Nonlinear Programming, 2nd ed. Athena Scientific, 1999.

[39] J.-B. Hiriart-Urruty and C. Lemaréchal, Fundamentals of convex analysis. Springer, 2001.

[40] J. Barzilai and J. M. Borwein, “Two-Point Step Size Gradient Methods,” IMA J. Num. Anal., vol. 8, no. 1, 1988.

[41] M. Schmidt, E. van den Berg, M. Friedlander, and K. Murphy, “Memory Projected Quasi-Newton Algorithm,” in International Conference on Artificial Intelligence and Statistics, 2009.

[42] E. G. Birgin, J. M. Martínez, and M. Raydan, “Algorithm 813: SPG - Software for Convex-constrained Optimization,” ACM Transactions on Mathematical Software, vol. 27, pp. 340–349, 2001.

[43] Y. Pang, Y. Yuan, and X. Li, “Gabor-based region covariance matrices for face recognition,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no. 7, pp. 989–993, 2008.

[44] M. T. Harandi, M. Salzmann, and R. Hartley, “From manifold to manifold: Geometry-aware dimensionality reduction for spd matrices,” in European Conference on Computer Vision. Springer, 2014.

[45] K. Lai, L. Bo, X. Ren, and D. Fox, “A large-scale hierarchical multi-view RGB-D object dataset,” in International Conference on Robotics and Automation, 2011.

[46] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” Pattern recognition, vol. 29, no. 1, pp. 51–59, 1996.

[47] A. Ess, B. Leibe, and L. V. Gool, “Depth and appearance for mobile scene analysis,” in International Conference on Computer Vision. IEEE, 2007.

[48] T. H. Lior Wolf and I. Maoz, “Face recognition in unconstrained videos with matched background similarity,” in Computer Vision and Pattern Recognition. IEEE, 2011.