Mach Learn (2017) 106:493–522
DOI 10.1007/s10994-016-5605-5

Geometry-aware principal component analysis for symmetric positive definite matrices

Inbal Horev[1] · **Florian Yger**[2] · **Masashi Sugiyama**[1]

Received: 15 February 2016 / Accepted: 24 October 2016 / Published online: 18 November 2016

Abstract Symmetric positive definite (SPD) matrices in the form of covariance matrices, for example, are ubiquitous in machine learning applications. However, because their size grows quadratically with respect to the number of variables, high-dimensionality can pose a difficulty when working with them. So, it may be advantageous to apply to them dimensionality reduction techniques. Principal component analysis (PCA) is a canonical tool for dimensionality reduction, which for vector data maximizes the preserved variance. Yet, the commonly used, naive extensions of PCA to matrices result in sub-optimal variance retention. Moreover, when applied to SPD matrices, they ignore the geometric structure of the space of SPD matrices, further degrading the performance. In this paper we develop a new Riemannian geometry based formulation of PCA for SPD matrices that (1) preserves more data variance by appropriately extending PCA to matrix data, and (2) extends the standard definition from the Euclidean to the Riemannian geometries. We experimentally demonstrate the usefulness of our approach as pre-processing for EEG signals and for texture image classification.

Keywords Dimensionality reduction · PCA · Riemannian geometry · SPD manifold · Grassmann manifold

Editors: Geoff Holmes, Tie-Yan Liu, Hang Li, Irwin King and Zhi-Hua Zhou.

✉ Inbal Horev
inbal@ms.k.u-tokyo.ac.jp

Florian Yger
florian.yger@dauphine.fr

Masashi Sugiyama
sugi@k.u-tokyo.ac.jp

[1] Department of Complexity Science and Engineering, University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa-shi, Chiba 277-8561, Japan

[2] LAMSADE Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France

1 Introduction

Covariance matrices are used in a variety of machine learning applications. Three prominent examples are computer vision applications (Tuzel et al. 2006, 2008), brain imaging (Pennec et al. 2006; Arsigny et al. 2006; Dryden et al. 2009) and brain computer interface (BCI) (Barachant et al. 2010, 2013) data analysis. In computer vision, covariance matrices in the form of region covariances are used in tasks such as texture classification. For brain imaging, the covariance matrices are diffusion tensors extracted from a physical model of the studied phenomenon. Finally, in the BCI community correlation matrices between different sensor channels are used as discriminating features for classification.

As discussed in Fletcher et al. (2004) and Harandi et al. (2014a), dimensionality can pose difficulties when working with covariance matrices because their size grows quadratically w.r.t. the number of variables. To deal with this issue, it is useful to apply to them dimensionality reduction techniques. A simple, commonly used technique is principal component analysis (PCA) (Jolliffe 2002). However, as we later show in Sect. 2, while vector PCA is optimal in terms of preserving data variance, the commonly used naive extensions of vector PCA to the matrix case (Yang et al. 2004; Lu et al. 2006) are sub-optimal for SPD matrices. Furthermore, when applied to SPD matrices, we argue that the standard Euclidean formulation of PCA disregards the inherent geometric structure of this space. We provide an in-depth review and discussion of the geometry of $\mathcal{S}_+^n$ and its advantages in Sect. 3. For the moment consider a brief motivation for the use of Riemannian geometry for dimensionality reduction of SPD matrices.

1.1 A case for the use of Riemannian geometry

The set $\mathcal{S}_+^n$ of symmetric positive definite (SPD) matrices of size $n \times n$, when equipped with the Frobenius inner product $\langle A, B\rangle_{\mathcal{F}} = \mathrm{tr}(A^\top B)$, belongs to a Euclidean space. A straightforward approach for measuring similarity between SPD matrices could be to use the Euclidean distance derived from the Euclidean norm. This is readily seen for 2×2 SPD matrices. A matrix $A \in \mathcal{S}_+^2$ can be written as $A = \left[\begin{smallmatrix} a & c \\ c & b \end{smallmatrix}\right]$ with $ab - c^2 > 0, a > 0$ and $b > 0$. Then matrices in $\mathcal{S}_+^2$ can be represented as points in $\mathbb{R}^3$ and the constraints can be plotted as an open convex cone whose interior is populated by SPD matrices (see Fig. 1). In this representation, the Euclidean geometry of symmetric matrices then implies that distances are computed along straight lines, shown as blue dashed lines in the figure.

In practice, however, the Euclidean geometry is often inadequate to describe SPD matrices extracted from real-life applications, e.g., covariance matrices. This observation has been discussed in Sommer et al. (2010). We observe similar behavior, as illustrated in Fig. 2. In this figure, we computed the vertical and horizontal gradients at every pixel of the image on the left. We then computed 2×2 covariance matrices between the two gradients for patches of pixels in the image. On the right, visualizing the same convex cone as in Fig. 1, every point represents a covariance matrix extracted from an image patch. The interior of the cone is not uniformly populated by the matrices, suggesting that they exhibit some structure that is not captured by the use of the straight geodesics of Euclidean geometry.

More generally, the use of Euclidean geometry for SPD matrices is known to generate several artifacts that may make it unsuitable for handling these matrices (Fletcher et al. 2004; Arsigny et al. 2007; Sommer et al. 2010). For example, for the simple task of averaging two matrices it may occur that the determinant of the average is larger than any of the two matrices. This effect is a consequence of the Euclidean geometry and is referred to as the

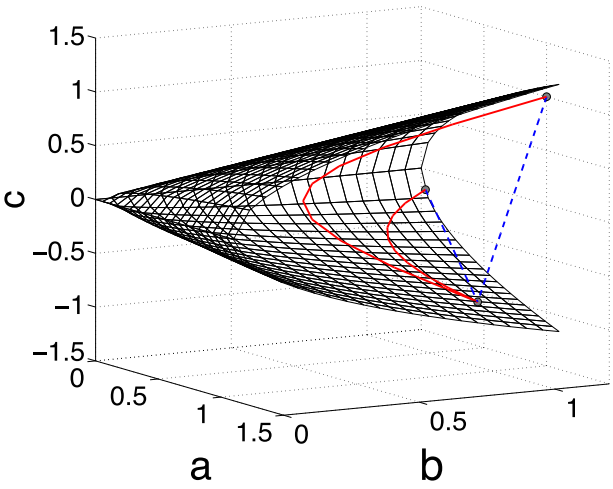

Fig. 1 Comparison between Euclidean (*blue straight dashed lines*) and Riemannian (*red curved solid lines*) distances measured between points of the space $\mathcal{S}_+^2$ (Color figure online)

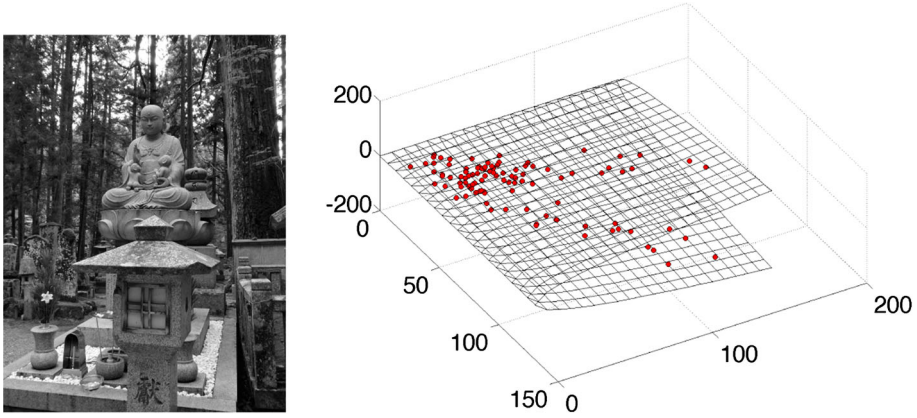

Fig. 2 Original image (*left*) and 2×2 covariance matrices between the image gradients, extracted from random patches of the image (*right*). *In the right plot*, the mesh represents the border of the cone of positive semi-definite matrices. The *image on the left* is courtesy of Bernard-Brunel and Dumont

swelling effect by Arsigny et al. (2007). It is particularly harmful for data analysis as it adds spurious variation to the data.

As a related example, take the computation of the maximum likelihood estimator of a covariance matrix, the sample covariance matrix (SCM). It is well known that with the SCM, the largest eigenvalues are overestimated while the smallest eigenvalues are underestimated (Mestre 2008). Since the SCM is an average of rank 1 symmetric positive semi-definite matrices, this may be seen as another consequence of the swelling effect.

Another drawback, illustrated in Fig. 1 and documented by Fletcher et al. (2004), is the fact that this geometry forms a non-complete space. This means that the extension of Euclidean geodesics is not guaranteed to stay within the manifold. Hence, interpolation

between SPD matrices is possible, but extrapolation may produce indefinite matrices, leading to uninterpretable solutions.

Contrarily, geodesics computed using the affine invariant Riemannian metric (AIRM) (Bhatia 2009) discussed thoroughly in Sect. 3.1, are shown in Fig. 1 as curved red lines. Their extension asymptotically approaches the boundary of the cone, but remains within the manifold.

Given the potentially erroneous results obtained by the Euclidean geometry, it would be disadvantageous to use this geometry to retain the principal modes of variation. A natural approach to cope with this issue is then to consider a Riemannian formulation of PCA. And, in fact, the development of geometric methods for applications involving SPD matrices has been a growing trend in recent years. In light of the rich geometric structure of $\mathcal{S}_+^n$, the disadvantages of the Euclidean geometry and the advances in manifold optimization, recently tools such as kernels (Barachant et al. 2013; Yger 2013; Jayasumana et al. 2013; Harandi et al. 2012) and divergences (Sra 2011; Cherian et al. 2011; Harandi et al. 2014b; Cichocki et al. 2014), as well as methods such as dictionary learning (Ho et al. 2013; Cherian and Sra 2014; Harandi and Salzmann 2015), metric learning (Yger and Sugiyama 2015), clustering (Goh and Vidal 2008; Kusner et al. 2014) and dimensionality reduction (Fletcher et al. 2004; Harandi et al. 2014a) have all been extended for SPD matrices using the Riemannian geometry.

1.2 Dimensionality reduction on manifolds

In statistics, such an extension of the PCA to a Riemannian setting has been studied for other manifolds. For example, it has been shown in Huckemann et al. (2010) for shape spaces that a Riemannian PCA was able to extract relevant principal components, especially in the regions of high curvature of the space where Euclidean approximation failed to appropriately explain data variation.

For the space of SPD matrices, a Riemannian extension of the PCA, namely the principal geodesic analysis (PGA), has been proposed in Fletcher et al. (2004). This algorithm essentially flattens the manifold at the center of mass of the data by projecting every element from the manifold to the tangent space at the Riemannian mean. In this Euclidean space a classical PCA is then applied. Although this approach is generic to any manifold it does not fully make use of the structure of the manifold, as a tangent space is only a *local* approximation of the manifold.

In this paper, we propose new formulations of PCA for SPD matrices. Our contribution is twofold: First and foremost, we adapt the basic formulation of PCA to make it suitable to matrix data and as a result capture more of the data variance. We tackle this problem in Sect. 2. Secondly, we extend PCA to Riemannian geometries to derive a truly Riemannian PCA which takes into account the curvature of the space and preserves the global properties of the manifold. This topic is covered in Sect. 3. More specifically, using a matrix transformation first used in Harandi et al. (2014a), we derive an unsupervised dimensionality reduction method maximizing a generalized variance of the data on the manifold. Section 4 is dedicated to a discussion of why our optimization problem is actually an approximation of an exact variance maximization objective, but we will show that it is sufficient for our purposes. Through experiments on synthetic data and real life applications, we demonstrate the efficacy of our proposed dimensionality reduction method. The experimental results are brought in Sect. 5.

2 Variance maximizing PCA for SPD matrices

In the introduction we discussed the need for dimensionality reduction methods for SPD matrices as well as the shortcomings of the commonly used PCA for this task. Essentially, although PCA is a dimensionality reduction method that *for vectors* optimally preserves the data variance, we show that its naive extensions to matrices do not do so optimally for SPD matrices.

In this section we tackle the problem of defining a PCA suitable for matrices. We begin by stating a formal definition of our problem. Next we show how to properly extend PCA from vectors to matrices so that we retain more of the data variance.

2.1 Problem setup

Let $\mathcal{S}_+^n = \left\{A \in \mathbb{R}^{n\times n} |\ \forall x \neq 0, x \in \mathbb{R}^n,\ x^\top A x > 0,\ A = A^\top\right\}$ be the set of all $n \times n$ SPD matrices, and let $\mathbf{X} = \{X_i \in \mathcal{S}_+^n\}_{i=1}^N$ be a set of N instances in $\mathcal{S}_+^n$. We assume that these matrices have some underlying structure, whereby their informative part can be described by a more compact, lower dimensional representation. Our goal is to compress the matrices, mapping them to a lower dimensional manifold $\mathcal{S}_+^p$ where $p < n$. In the process, we wish to keep only the relevant part while discarding the extra dimensions due to noise.

The task of dimensionality reduction can be formulated in two ways: First, as a problem of minimizing the residual between the original matrix and its representation in the target space. Second, it can be stated in terms of variance maximization, where the aim is to find an approximation to the data that accounts for as much of its variance as possible. In a Euclidean setting these two views are equivalent (Bishop 2007, Chap. 12). However, in the case of SPD matrices, $\mathcal{S}_+^p$ is not a sub-manifold of $\mathcal{S}_+^n$ and elements of the input space cannot be directly compared to elements of the target space. Thus, focusing on the second view, we search for a mapping $\mathcal{S}_+^n \mapsto \mathcal{S}_+^p$ that best preserves the Fréchet variance σ_δ^2 of $\mathbf{X}$, defined below.

Following the work of Fréchet (1948) we define σ_δ^2 via the Fréchet mean, which is unique in our case.

Definition 1 (*Fréchet Mean*) The (sample) Fréchet mean of the set $\mathbf{X}$ w.r.t. the metric δ is

$$\bar{X}_\delta = \underset{X\in\mathcal{S}_+^n}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} \delta^2\left(X_i, X\right).$$

Then, the Fréchet variance is:

Definition 2 (*Fréchet Variance*) The (sample) Fréchet variance of the set $\mathbf{X}$ w.r.t. the metric δ is given by

$$\sigma_\delta^2 = \frac{1}{N}\sum_{i=1}^{N} \delta^2\left(X_i, \bar{X}_\delta\right).$$

As in Harandi et al. (2014a), we consider for any matrix $X \in \mathcal{S}_+^n$ a mapping to $\mathcal{S}_+^p$ (with $p < n$) parameterized by a matrix $W \in \mathbb{R}^{n\times p}$ which satisfies the orthogonality constraint $W^\top W = \mathbb{I}_p$, where $\mathbb{I}_p$ is the $p \times p$ identity matrix. The mapping then takes the form of $X^\downarrow = W^\top X W$.

2.2 Variance maximization with gaPCA and PCA

Having framed the optimization of the matrix W in terms of maximization of the data variance our proposed formulation, explicitly written in terms the Fréchet variance, is:

Definition 3 (*Geometry-aware PCA (gaPCA)*) gaPCA is defined as

$$W = \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta^2 \left(W^\top X_i W, W^\top \bar{X}_\delta W \right), \tag{1}$$

where $\mathcal{G}(n, p)$ is the Grassmann manifold, the set of all p-dimensional linear subspaces of $\mathbb{R}^n$.

In reality, the formulation above does not *exactly* express the Fréchet variance of the compressed set $\mathbf{X}^\downarrow$, but expresses it only approximately. For now we simply state this fact without further explanation and continue with the exposition and analysis. We dedicate Sect. 4 to a detailed investigation of this issue.

As described in Edelman et al. (1998) and Absil et al. (2009), an optimization problem such as the one in Definition 3 can be formulated and solved on either the Stiefel or the Grassmann manifold. The use of the Stiefel manifold, the set of p-rank matrices in $\mathbb{R}^{n\times p}$ with orthogonal columns, would impose the necessary orthogonality constraint on the transformation matrix W. However, note that for our problem the individual components are of little importance to us. Our interest is in reducing the size of the input matrices.[1] As such, each single component yields a 1×1 matrix which is not very informative in it of itself. Rather, we are interested in the principal components as an ensemble, i.e., the p-dimensional linear space that they span. Rotation within the p-dimensional space will not affect our solution. Thus, it suffices to consider a mapping $X^\downarrow = W^\top X W$ with $W \in \mathcal{G}(n, p)$.

We compare our variance-based definition to PCA, the canonical method for dimensionality reduction which itself aims to preserve maximal data variance. For vector data, PCA is formulated as

$$\begin{aligned} W &= \underset{W^\top W = \mathbb{I}_p}{\operatorname{argmax}} \sum_i \| (x_i - \bar{x})\, W \|_2^2 \\ &= \underset{W^\top W = \mathbb{I}_p}{\operatorname{argmax}} \operatorname{tr} \left(W^\top \left(\sum_i (x_i - \bar{x})^\top (x_i - \bar{x}) \right) W \right), \end{aligned} \tag{2}$$

where $\bar{x}$ is the Euclidean mean of the data.

Translating the operations in the right-most formulation of Eq. (2) from the vector case to the matrix case gives

$$W = \underset{W^\top W = \mathbb{I}_p}{\operatorname{argmax}} \operatorname{tr} \left(W^\top \left(\sum_i \left(X_i - \bar{X}_\mathrm{e} \right)^\top \left(X_i - \bar{X}_\mathrm{e} \right) \right) W \right), \tag{3}$$

where $\bar{X}_\mathrm{e}$ is the Euclidean mean of the data.

For symmetric matrices, this formulation is equivalent to the one proposed in Yang et al. (2004) and Lu et al. (2006). Note, however, that the matrix W in Eq. (3) acts on the data only by right-hand multiplication. Effectively, it is as if we are performing PCA only on the row space of the data matrices $\mathbf{X}$.[2]

Indeed, the key difference between our proposed method and ordinary PCA is that in our cost function the matrix W acts on $\mathbf{X}$ on both sides. It is only by applying W to both sides of a matrix that we obtain a valid element in $\mathcal{S}_+^p$. So, it is only natural that W should also act on both its sides during optimization.

[1] And not in a low rank approximation of the same size.

[2] In our case the matrices are symmetric so PCA on the row space and on the column space are identical.

Although our method can accommodate multiple geometries via various choices of the metric δ, the difference between Eqs. (3) and (1) becomes apparent when we work, as the standard PCA does, in the Euclidean geometry. In the Euclidean case, the cost function optimization in Definition 3 becomes

$$\begin{aligned} W &= \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \left\| W^\top \left(X_i - \bar{X}_\mathrm{e}\right) W \right\|_{\mathcal{F}}^2 \\ &= \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \operatorname{tr}\left(W^\top \left(X_i - \bar{X}_\mathrm{e}\right)^\top W W^\top \left(X_i - \bar{X}_\mathrm{e}\right) W \right). \end{aligned} \tag{4}$$

Note the additional term $WW^\top \neq \mathbb{I}_n$ as compared to Eq. (3).

In general, the two expressions are not equivalent. Moreover, our proposed formulation consistently retains more of the data variance than the standard PCA method, as will be shown in Sect. 5. It is worth noting, however, that the two methods are equivalent when the matrices $\{X_i\}$ share the same eigenbasis. In this case, the problem can be written in terms of the common basis and the extra term $WW^\top$ does not contribute to the cost function. Both methods then yield identical results.

3 Geometry-aware PCA

Equipped with a PCA formulation adequate for matrix data, we now turn to the issue of geometry awareness. We have already discussed that the use of a Euclidean geometry may potentially lead to erroneous or distorted results when applied to SPD matrices, especially when the distance between matrices is large on the manifold.[3] Other methods for dimensionality reduction of SPD matrices, while utilizing the structure of the SPD manifold, are nonetheless flawed. First, they use only a local approximation of the manifold, causing a degradation in performance when distances between matrices are large. Second, and more importantly, these methods apply the same faulty formulation of matrix PCA.

In the previous section we presented a general definition of gaPCA in Definition 3. In order to highlight the difference between PCA and gaPCA we studied its instantiation with the Euclidean geometry. However, this formulation can incorporate any metric defined on $\mathcal{S}_+^n$. In this section we explore these other metrics.

We first describe the geometry of the SPD matrix manifold. We will highlight the benefits of our geometry-aware method through a discussion of some of the relevant invariance properties of the various metrics defined on the manifold. These properties will allow for efficient optimization of the transformation matrix W. Finally, we present some useful implementation details.

3.1 The geometry of $\mathcal{S}_+^n$

The shortcomings of the Euclidean geometry described in the introduction stem from the fact that it does not enforce the positive-definiteness constraint. That is, using the Eucliden metric we are essentially treating the matrices as if they were elements of the (flat) space $\mathbb{R}^{n\times n}$. Nonetheless, the positive-definiteness constraint induces a Riemannian manifold of negative curvature. As noted in Fletcher et al. (2004) and Sommer et al. (2010), it is then

[3] Errors also occur because the sample eigenvectors, i.e., the principal components, are sensitive even to small perturbations and are, as a result, rarely correctly estimated (Anderson 1963). However, this is also the case for other geometries.

more natural to use a Riemannian geometry as it ensures that the solutions will respect the constraint encoded by the manifold.

A standard choice of metric, due to its favorable geometric properties, is the affine invariant Riemannian metric (AIRM) (Bhatia 2009). In this paper we denote it as δ_{r}. Equipped with this metric, the SPD manifold becomes a complete manifold of negative curvature. It prevents the occurrence of the swelling effect and allows for matrix extrapolation without obtaining non-definite matrices. Other beneficial properties that are specifically related to our problem are presented later in this section.

Definition 4 (*Affine invariant Riemannian metric (AIRM)*) Let $X, Y \in \mathcal{S}_+^n$ be two SPD matrices. Then, the AIRM is given as

$$\delta_{\mathrm{r}}^2(X, Y) = \left\| \log\left(X^{-1/2} Y X^{-1/2}\right) \right\|_{\mathcal{F}}^2 = \left\| \log\left(X^{-1} Y\right) \right\|_{\mathcal{F}}^2,$$

where $\log(\cdot)$ is the matrix logarithm function, which for SPD matrices is $\log(X) = U \log(\Lambda) U^\top$ for the eigendecompostion $X = U \Lambda U^\top$.

Another noteworthy metric, closely related to the AIRM, is the log-determinant (a.k.a symmetric Stein) metric (Sra 2011) which we denote by δ_{s}.

Definition 5 (*Log-determinant (symmetric Stein) metric*) Let $X, Y \in \mathcal{S}_+^n$ be two SPD matrices. Then, the log-determinant (symmetric Stein) metric is given as

$$\delta_{\mathrm{s}}^2(X, Y) = \log\left(\det\left((X + Y)/2\right)\right) - \log\left(\det(XY)\right)/2.$$

This metric is dubbed 'symmetric' because it is a symmetrized Bregman matrix divergence (Cherian et al. 2011; Sra 2012). These in turn have been shown to be useful for various learning applications (Cherian et al. 2011; Harandi et al. 2014b). While not a Riemannian metric, the log-determinant metric closely approximates the AIRM and shares many of its useful properties. Furthermore, in Sra (2011) it has been suggested that the log-determinant metric is a more computationally efficient alternative to the AIRM. The advantageous properties of this metric relevant to our problem are shared by both δ_{s} and δ_{r} and described later in this section.

Even though $\mathcal{S}_+^n$ is a curved space, flat subspaces (briefly, *flats*) (Bridson and Haefliger 2011) are a powerful tool with which to study its structure and that of other symmetric spaces. Formally, a subspace $F \subset \mathcal{S}_+^n$ is called a flat of dimension k (a *k-flat*) if it is isometric to $\mathbb{E}^k$, the Euclidean space of dimension k. If F is not contained in any flat of bigger dimension, then F is called a maximal flat. Next we give a functional definition of flats of $\mathcal{S}_+^n$.

Definition 6 (*Flats of $\mathcal{S}_+^n$*) Let $A \in S(n)$, where $S(n)$ is the space of symmetric matrices. Then, consider elements $f \in \mathcal{S}_+^n$ that share the eigenvectors Q with e^A, the matrix exponential of A. These elements all commute with each other and with e^A. We call this space the n-flat F.

Since the elements of F commute with each other, i.e., $UV = VU$ for $U, V \in F$, we have $\log(UV) = \log(U) + \log(V)$ and the matrix log log $(\cdot)$. So, the distance between them is

$$\delta(U, V) = \sqrt{(\mathrm{tr}(\log(U^{-1}V)^2))} = \sqrt{(\mathrm{tr}((\log(V) - \log(U))^2))}. \tag{5}$$

Since $\sqrt{(\mathrm{tr}(\cdot)^2)}$ is a Euclidean norm, we have that F is isomorphic to $\mathbb{R}^n$ with a Euclidean metric under $\log(\cdot)$.

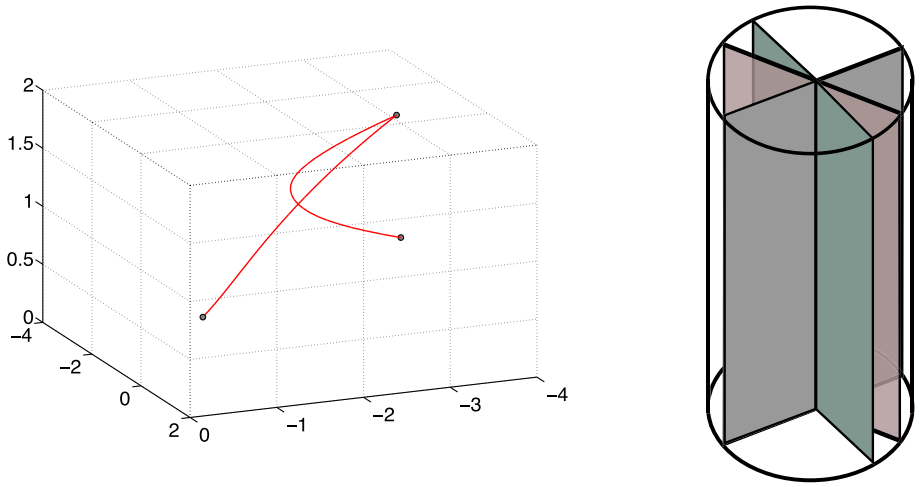

Fig. 3 Riemannian geodesics plotted in cylindrical coordinates (*left*). Flats in $\mathcal{S}_+^2$ shown as panes of a revolving door (*right*)

Note that $\delta(U, V) = \sqrt{(\text{tr}((\log(V) - \log(U))^2))}$ is precisely the log-Euclidean metric (Arsigny et al. 2007).

Definition 7 (*Log-Euclidean metric*) Let $X, Y \in \mathcal{S}_+^n$ be two SPD matrices. Then, the log-Euclidean metric is given as

$$\delta_{\text{le}}^2(X, Y) = \|\log(X) - \log(Y)\|_{\mathcal{F}}^2 .$$

As illustrated in Yger and Sugiyama (2015), this metric uses the logarithmic map to project the matrices to $\mathcal{T}_{\mathbb{I}}\mathcal{S}_+^n$, the tangent space at the identity, where the standard Euclidean norm is then used to measure distances between matrices. So, the use of this metric in effect flattens the manifold, deforming the distance between matrices when they do not lie in the same n-flat.

In the context of this work, flats provide a valuable intuitive understanding of our proposed method. This is because the goal of our geometry aware PCA method is to find an optimal transformation matrix Q, which is a p-flat. Viewing $\mathcal{S}_+^n$ in terms of flats is especially instructive for $\mathcal{S}_+^2$ where the eigenvectors create a 2-dimensional rotation matrix that can be summed up by the rotation angle θ. Then, the matrices in $\mathcal{S}_+^2$ can be plotted in cylindrical coordinates: θ is the rotation angle, ρ is ratio of eigenvalues and $z = \log\det X$ as seen in Fig. 3a. In this figure geodesics between matrices, computed using the AIRM, are shown in red. With this construction, flats of $\mathcal{S}_+^2$ can be imagined as panes of a revolving door, intersecting at the line formed by multiples of the identity $\alpha\mathbb{I}$, for $\alpha > 0$ (see Fig. 3b). In Sect. 5 we will use flats to measure the quality of the eigenspace estimation as compared to the true maximal variance eigenspace.

3.2 Invariance properties

As discussed in Bhatia (2009), δ_{r} is invariant under the congruence transformation $X \mapsto P^\top X P$ for $P \in GL_n(\mathbf{R})$, the group of $n \times n$ real valued invertible matrices. So, we have

$$\delta_{\mathrm{r}}(X, Y) = \delta_{\mathrm{r}}(P^{\top} X P, P^{\top} Y P)$$

for $X, Y \in \mathcal{S}_{+}^{n}$ and $P \in GL_n(\mathbf{R})$. This property is called affine invariance and is also shared by δ_{s} (Sra 2012). For brevity we will state the subsequent analysis in terms of the AIRM, but the same will hold true for the log-determinant metric.

The affine invariance property has practical consequences for covariance matrices extracted from EEG signals (Barachant and Congedo 2014). Indeed, such a class of transformations includes re-scaling and normalization of the signals, electrode permutations and, if there is no dimensionality reduction, it also includes whitening, spatial filtering or source separation. For covariance matrices extracted from images this property has similar implications, with this class of transformations including changes of illumination when using RGB values (Harandi et al. 2014a).

Contrarily, the log-Euclidean metric and the distance derived from it are not affine-invariant. This fact has been used to derive a metric learning algorithm (Yger and Sugiyama 2015). Nevertheless, it is invariant under the action of the orthogonal group. This comes from the fact that for any SPD matrix X and invertible matrix P, we have $\log(PXP^{-1}) = P \log(X) P^{-1}$ (Bhatia 2009, p. 219). Then, using the fact that for any matrix $O \in \mathbb{O}_p$, $O^{\top} = O^{-1}$, it follows that $\delta_{\mathrm{le}}(O^{\top} X O, O^{\top} Y O) = \delta_{\mathrm{le}}(X, Y)$.

3.3 Optimization on manifolds

Our approach may be summed up as finding a lower-dimensional manifold $\mathcal{S}_{+}^{p}$ by optimizing a transformation (parameterized by W) that maximizes the (approximate) Fréchet variance w.r.t. a metric δ. As the parameter W lies in the Grassmann manifold $\mathcal{G}(n, p)$, we solve the optimization problem on this manifold (Absil et al. 2009; Edelman et al. 1998).

Optimization on matrix manifolds is a mature field and by now most of the classical optimization algorithms have been extended to the Riemannian setting. In this setting, descent directions are not straight lines but rather curves on the manifold. For a function f, applying a Riemannian gradient descent can be expressed by the following steps, illustrated in Fig. 4:

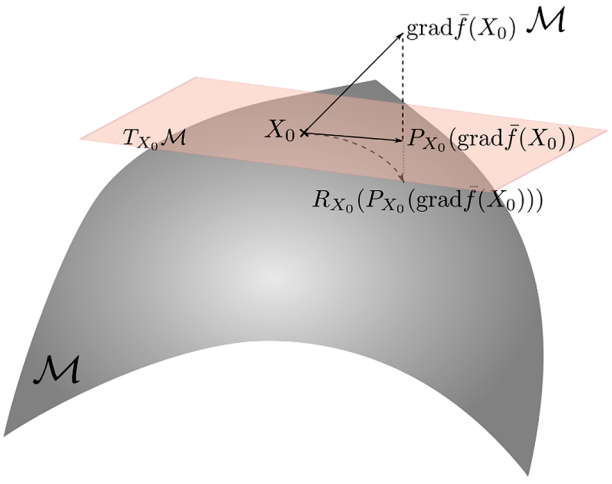

Fig. 4 Function optimization on manifolds. The Euclidean gradient of the function $\bar{f}$ is projected onto the tangent space at the point X_0, then onto the manifold using the exponential map or a retraction

1. At any iteration, at the point W, transform a Euclidean gradient $D_W f$ into a Riemannian gradient $\nabla_W f$. In our case, i.e., for the Grassmann manifold, the transformation is $\nabla_W f = D_W f - WW^\top D_W f$ (Absil et al. 2009).
2. Perform a line search along geodesics at W in the direction $H = \nabla_W f$. For the Grassmann manifold, on the geodesic going from a point W in direction H (with a scalar step-size $t \in \mathbb{R}$), a new iterate is obtained as $W(t) = WV\cos(\Sigma t)V^\top + U\sin(\Sigma t)V^\top$, where $U\Sigma V^\top$ is the compact singular value decomposition of H and the matrix sine and cosine functions are defined by $\cos(A) = \sum_{k=0}^{\infty}(-1)^k \frac{A^{2k}}{2k!}$ and $\sin(A) = \sum_{k=0}^{\infty}(-1)^k \frac{A^{2k+1}}{(2k+1)!}$.

In practice, we employ a Riemannian trust-region method described in Absil et al. (2009) and efficiently implemented in Boumal et al. (2014).

3.4 Cost function gradients

In order to make this article self-contained, we provide the Euclidean gradient of our cost function w.r.t. W for each of the four metrics described above. For the Euclidean metric δ_{e} this is:

$$D_W \delta_{\mathrm{e}}^2(W^\top X_i W, W^\top \bar{X}_{\mathrm{e}} W) = 4\left(X_i - \bar{X}_{\mathrm{e}}\right) W W^\top \left(X_i - \bar{X}_{\mathrm{e}}\right) W.$$

Its derivation is detailed in "Appendix 1".

Then, for the AIRM δ_{r} we have, following Harandi et al. (2014a).

$$\begin{aligned} &D_W \delta_{\mathrm{r}}^2 \left(W^\top X_i W, W^\top \bar{X}_{\mathrm{r}} W\right) \\ &= 4\left(X_i W \left(W^\top X W\right)^{-1} - \bar{X}_{\mathrm{r}} W \left(W^\top \bar{X}_{\mathrm{r}} W\right)^{-1}\right) \log\left(W^\top X_i W \left(W^\top \bar{X}_{\mathrm{r}} W\right)^{-1}\right). \end{aligned}$$

It should be noted that using directional derivatives (Bhatia 1997; Absil et al. 2009) we obtain a different (but numerically equivalent) formulation of this gradient. For completeness, we report this formula and its derivation in "Appendix 2" of the supplementary material. In our experiments, as it was computationally more efficient, we use the equation above for the gradient.

Next, the gradient w.r.t. W of the log-determinant metric δ_{s} is given in Harandi et al. (2014a) by:

$$\begin{aligned} D_W \delta_{\mathrm{s}}^2 \left(W^\top X_i W, W^\top \bar{X}_{\mathrm{r}} W\right) = \left(X_i + \bar{X}_{\mathrm{r}}\right) W \left(W^\top \frac{X_i + \bar{X}_{\mathrm{r}}}{2} W\right)^{-1} \\ - X_i W \left(W^\top X_i W\right)^{-1} - \bar{X}_{\mathrm{r}} W \left(W^\top \bar{X}_{\mathrm{r}} W\right)^{-1}. \end{aligned}$$

Finally, For the log-Euclidean metric:

$$\begin{aligned} &D_W \delta_{\mathrm{le}}^2(W^\top X_i W, W^\top \bar{X}_{\mathrm{le}} W) \\ &= 4\left(X_i W D \log\left(W^\top X_i W\right)\left[\log\left(W^\top X_i W\right) - \log\left(W^\top \bar{X}_{\mathrm{le}} W\right)\right]\right. \\ &\quad \left. + \bar{X}_{\mathrm{le}} W D \log\left(W^\top \bar{X}_{\mathrm{le}} W\right)\left[\log\left(W^\top \bar{X}_{\mathrm{le}} W\right) - \log\left(W^\top X_i W\right)\right]\right), \end{aligned}$$

where $Df(W)[H] = \lim_{h\to 0} \frac{f(W+hH)-f(W)}{h}$ is the Fréchet derivative (Absil et al. 2009) and $\bar{X}_{\mathrm{le}}$ denoted the mean w.r.t. the log-Euclidean metric. Note that there is no closed-form solution for $D\log(W)[H]$ but it can be numerically computed efficiently (Boumal 2010; Boumal and Absil 2011; Al-Mohy and Higham 2009). This derivation is given in "Appendix 3".

4 Exact variance maximization

Briefly stated in Sect. 2, we now explain why Eq. (1) constitutes only an approximation to the Fréchet variance maximization. In general, the operations of computing the mean and projecting onto the lower dimensional manifold are not interchangeable. That is, let $\bar{X}^{\downarrow}$ be the mean of the compressed set $\mathbf{X}^{\downarrow} = \left\{W^{\top} X_i W\right\}$ and let $W^{\top} \bar{X} W$ be the compressed mean of the original set $\mathbf{X}$. The two matrices are not equal in general for metrics other than the Euclidean metric. This is because for the δ_{r}, δ_{s} and δ_{le} the Fréchet mean of the set $\mathbf{X}$ is not a linear function of the matrices. Since we do not know in advance the mean of the compressed set, the cost function defined in Eq. (1) does not *exactly* express the Fréchet variance of $\mathbf{X}^{\downarrow}$. Rather, it serves as an approximation to it.

In the following we bring two strategies to address this issue. First, we propose to whiten the input matrices before optimizing the transformation matrix W. Pre-whitening ensures that the Riemannian mean of the set $\mathbf{X}$ is known and equal to the identity $\bar{X}_{\mathrm{r}} = I$. Second, we consider a constrained formulation of the cost function, where the constraint ensures that we are using the mean of the compressed set $\mathbf{X}^{\downarrow}$. The constrained formulation is brought here for the AIRM, but a similar technique can also be applied to the other metrics.

Ultimately, we find that the relaxed optimization problem in Definition 3 is sufficient for our purposes. Although feasible, our experiments will show that there is no practical benefit in solving the exact variance maximization problem. Nonetheless, the two methods discussed below highlight interesting aspects of the problem that can be applied to other optimization problems based on the distance between SPD matrices.

4.1 Pre-whitening the data

Our first potential strategy is to pre-whiten the data before optimizing over W. Using the Riemannian geometry, each point is mapped to $X_i \mapsto \tilde{X}_i = \bar{X}_{\mathrm{r}}^{-1/2} X_i \bar{X}_{\mathrm{r}}^{-1/2}$. Subsequently, the Riemannian mean of $\tilde{\mathbf{X}}$ is the identity $\mathbb{I}_n$. Using the whitened data, the resulting cost function is

$$W = \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2 \left(W^{\top} \tilde{X}_i W, \mathbb{I}_p\right). \tag{6}$$

Due to the affine invariance property of the AIRM and the log-determinant metric, we have that $\delta\left(\tilde{X}_i, \mathbb{I}\right) = \delta\left(X, \bar{X}\right)$. That is, for these two metrics the whitened data is simply a translated and rotated copy of the original input data with the distances between matrices preserved. Thus, up to rotation and scaling, we should expect the solution of this problem to be equivalent to that of the original formulation in Definition 3 in terms of retained variance.

4.2 A constrained optimization problem

To obtain an exact variance maximization problem we may use the following fact that the Riemannian mean $\bar{X}_{\mathrm{r}}$ of a set $\mathbf{X} = \{X_i \in \mathcal{S}_+^n\}_{i=1}^N$ can be shown to satisfy the following equation (Bhatia 2013):

$$C\left(W, \bar{X}_{\mathrm{r}}\right) = \sum_{j=1}^{n} \log\left(\left(W^{\top} X_j W\right)^{-1/2} \bar{X}_{\mathrm{r}} \left(W^{\top} X_j W\right)^{-1/2}\right) = 0^{p \times p}. \tag{7}$$

This is a direct consequence of the fact that the Riemannian mean uniquely minimizes the Fréchet variance $\sum_{j=1}^{n} \delta_{\mathrm{r}}^2 \left(W^{\top} X_j W, \bar{X}_{\mathrm{r}}\right)$. While this equation does not have a closed form

solution, we may use the implicit constraint $C\left(W, \bar{X}_{\mathrm{r}}\right) = 0^{p\times p}$ to formulate a cost function which exactly embodies the Fréchet variance of the data. By incorporating the relation between W and $\bar{X}_{\mathrm{r}}$ through the constraint, we ensure that at each step of the optimization we use the exact mean of $\mathbf{X}^{\downarrow}$, and not an approximation of it.

Consider the following constrained optimization problem, given by

$$\hat{f}(W) = f(W, \Lambda) = \sum_{j=1}^{n} \delta_{\mathrm{r}}^2 \left(W^{\top} X_j W, \Lambda\right) \text{ s.t. } C(W, \Lambda) = 0^{p\times p}, \tag{8}$$

where for brevity we leave out the dependence of f and C on $\mathbf{X}$. Through the use of the implicit function theorem (Krantz and Parks 2012), we are guaranteed the existence of a differentiable function $(W) : \mathcal{S}(n, p) \to \mathcal{S}_+^p$ defined by $C(W, \Lambda) = 0^{p\times p}$. Then, the derivative ∇_W can be used to express the gradient of the cost w.r.t. W:

$$\nabla \hat{f}(W) = \nabla_W (W) \nabla_{\Lambda} f(W, \Lambda) + \nabla_W f(W, \Lambda). \tag{9}$$

Since in this cost function the transformation matrix W is applied only to one argument of the distance function, our problem is no longer invariant to the action of the orthogonal group $\mathbb{O}_p$. So, we must optimize this cost over the Stiefel manifold.

Practically, for optimization of the constrained cost function, at each step we:

1. Given the current value of W, compute the Riemannian mean of the set $\mathbf{X}^{\downarrow}$.
2. Compute $\nabla \hat{f}(W)$ using Eq. (9) for use in our line search.

This method will be denoted as δ_{c} (for *constrained*). Details of the computation of the cost function gradient are brought in "Appendix 4".

5 Experimental results

To understand the performance of our proposed methods, we test them on both synthetic and real data. First, for synthetically generated data, we examine their ability to compress the data while retaining its variance. We also show that our methods are superior in terms of eigenspace estimation. Next, we apply them to image data in the form of region covariance matrices as well as to brain computer interface (BCI) data in the form of covariance matrices. To assess the quality of the dimensionality reduction, we use the compressed matrices for classification and examine the accuracy rates. Finally, we briefly compare the constrained cost formulation δ_{c} discussed in Sect. 4.2, which optimizes the exact Fréchet variance, to δ_{r}PCA, which uses only an approximate expression of the variance.

5.1 Synthetic data

Our first goal is to substantiate the claim that our methods outperform the standard matrix PCA in terms of variance maximization. As shown in Chapter 6 of Jolliffe (2002), it is useful to study the fraction of variance retained by the method as the dimension grows. To this end we randomly generate a set $\mathbf{X} = \left\{X_i \in \mathcal{S}_+^n\right\}$ of 50 SPD matrices of size $n = 17$ using the following scheme.

For each X_i, we first generate an $n \times n$ matrix A whose entries are i.i.d. standard normal random variables. Next, we compute the QR decomposition of this matrix $A = QR$, where Q is an orthonormal matrix and R is an upper triangular matrix. We use Q as the eigenvectors of X_i. Finally, we uniformly draw its eigenvalues $\lambda = (\lambda_1, \ldots, \lambda_n)$ from the range [0.5, 4.5].

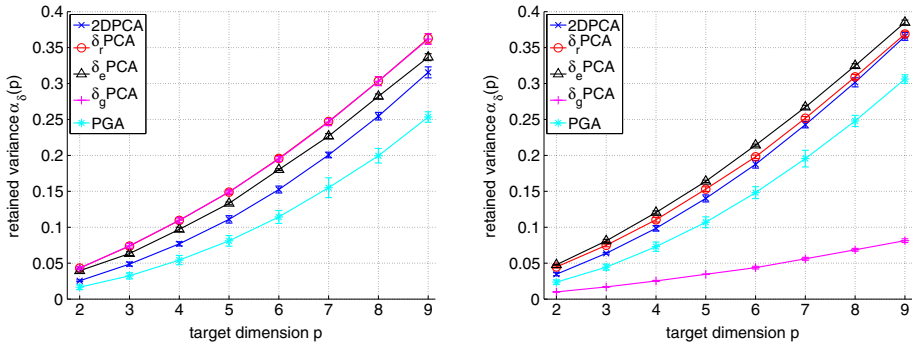

Fig. 5 Fraction of Fréchet variance retained by the various compression methods w.r.t. **a** the Riemannian and **b** the Euclidean distances. Metrics which obtained identical results to δ_{r} have been omitted from the figure for clarity. The *error bars* indicate the standard deviation

The resulting matrices are then $X_i = Q_i \mathrm{diag}\,(\lambda_i)\, Q_i^\top$, where each X_i has a unique matrix Q_i and spectrum λ_i.

Each matrix was compressed to size $p \times p$ for $p = 2, \ldots, 9$ using gaPCA with various metrics, 2DPCA (Yang et al. 2004) and PGA (Fletcher et al. 2004). PGA first maps the matrices $\mathbf{X}$ via the matrix logarithm to $\mathcal{T}_{\bar{X}_{\mathrm{r}}} \mathcal{S}_+^n$, the tangent space at the point $\bar{X}_{\mathrm{r}}$. Then standard linear PCA is performed in the (Euclidean) tangent space. The matrix W was initialized by the same random guess drawn from $\mathcal{G}\,(n,\, p)$ for each of the metrics of gaPCA. We recorded the fraction of the Fréchet variance contained in the compressed dataset for the various values of p,

$$\alpha_\delta\,(p) = \frac{\sigma_\delta^2\left(\mathbf{X}^{\downarrow}(p)\right)}{\sigma_\delta^2\,(\mathbf{X})},$$

for the Euclidean and Riemannian metrics. This process was repeated 25 times for different instances of the dataset $\mathbf{X}$.

The averaged results of the experiment, along with standard deviations, are presented in Fig. 5. For clarity, when the curves of various methods coincide, only one is kept, while the others are omitted. The methods δ_{s}PCA and δ_{le}PCA obtained nearly identical results to δ_{r}PCA. The same is true for the constrained optimization problem δ_{c}PCA. The subspace spanned by the columns of the transformation matrix W obtained by δ_{c}PCA is not identical to that of δ_{r}PCA, but the difference in the retained variance is negligible. This implies that $W^\top \bar{X}_{\mathrm{r}} W$ is a good approximation for the Riemannian mean of the compressed set $\mathbf{X}^{\downarrow}$. The curves of δ_{r}PCA and δ_{g}PCA coincide for the Riemannian variance since the AIRM is invariant to the process of data whitening.

We see that our proposed methods retain the greatest fraction of data variance. As expected, each gaPCA method is best at retaining the variance w.r.t. its own metric. That is, for the variance w.r.t. the AIRM, δ_{r}PCA outperforms δ_{e}PCA, and for the Euclidean metric the opposite is true. The only exception is δ_{g}PCA, which performs poorly w.r.t. the Euclidean variance. This is due to the data centering performed before the dimensionality reduction. Recall that the data centering is done using the Riemannian geometry, i.e., $\tilde{X}_i = \bar{X}_{\mathrm{r}}^{-1/2} X_i \bar{X}_{\mathrm{r}}^{-1/2}$. While this transformation preserves the Riemannian distance between matrices, it does not preserve the Euclidean distance. Thus, we obtain poor results for the Euclidean metric using this method.

Table 1 Average distance between the estimated p dimensional subspace and the true subspace F

p	5	7	8	10	11	13
2DPCA	2.53	2.69	2.65	2.445	2.46	1.9
PGA	**2.39**	**2.48**	**2.48**	2.44	2.24	1.84
δ_rPCA	3.12	2.87	2.72	2.06	**1.9**	**1.43**
δ_ePCA	3.22	3.04	2.61	**2**	1.93	1.55
δ_sPCA	3.12	2.87	2.65	2.0577	**1.9**	**1.43**
δ_{le}PCA	3.12	2.87	2.72	2.06	**1.9**	**1.43**
δ_cPCA	3.12	2.87	2.72	2.06	**1.9**	**1.43**
δ_gPCA	3.13	2.93	2.67	2.1	**1.9**	**1.43**

Best results are marked in boldface

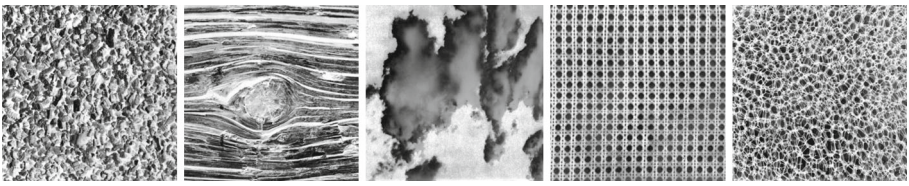

Fig. 6 Samples from the normalized Brodatz texture dataset

Next, we examine the quality of the eigenspace estimation. To this end, we generate data using the following scheme: We begin with a basis for $\mathbb{R}^n$. The first p vectors of this basis are denoted by F, and the remaining $n-p$ vectors are denoted by G. For each matrix X_i we randomly generate a $p \times p$ rotation matrix O_i and an $n \times n$ rotation matrix R_i. The input matrices are then given by $X_i = Q_i \text{diag}(\lambda_i) Q_i^\top$ where for $\epsilon \ll 1$, $Q_i = [F O_i \; G](\mathbb{I} + \epsilon R_i)$ and λ_i is a set of n eigenvalues uniformly drawn from the range $[0.5, 1.5]$. By rotating F via the matrix O_i we have that each $F O_i$ spans the same $n \times p$ space but lies in a different p-flat.[4] The matrix R_i acts as a small perturbation, slightly rotating the entire eigenbasis. Without the perturbation R_i, projecting onto the $n \times p$ space spanned by F would, in general, ensure maximal variance.[5]

We ran the experiment for a range of subspaces F of sizes $p = 5, 7, 8, 10, 11, 13$ and for a constant number of 15 eigenvectors in G. For each value of p computed the distance between the transformation matrix W obtained by each method and the space F. We repeated the experiment 25 times.

The results are brought in Table 1. We see that as p grows, our gaPCA methods manage to more accurately estimate the true eigenspace F as compared to the other methods.

5.2 Texture image data

Following the promising results on the synthetic data, we next test our methods on real data. In computer vision applications, region covariance (RC) matrices (Tuzel et al. 2006, 2008) are useful image descriptors, computed using the covariance between feature vectors taken from image patches. In this set of experiments we performed a texture classification task on the normalized Brodatz (1966) dataset, a set of 112 grayscale texture images (see Fig. 6).

[4] See Sect. 3.1 for a discussion of flats of $\mathcal{S}_+^n$.

[5] The maximal variance also depends on the values of the eigenvalues λ.

In each experiment we randomly selected two texture images. For each image the left side was used to create the training set and the right side was used for the test set. The RC matrices were created by extracting 128×128 patches from randomly selected locations in the image. Then, at every pixel the feature vectors were composed of 28 Gabor filter responses (Gabor 1946) for four scales and seven angles, as well as pixel intensity and first and second gradients, for a total of 34 feature dimensions. The RCs were compressed to size 8×8 using the standard PCA, PGA and our proposed gaPCA, then classified using two different classifiers—a nearest neighbor method and a minimum distance to the mean (MDM) (Barachant et al. 2012) scheme as follows: We first apply our methods in an *unsupervised* manner. Next, using the labels of the training set, we compute the mean for each of the two classes. Then, we classify the covariance matrices in the test set according to their distance to the class means; each test covariance matrix is assigned the class to which it is closer. This classifier is restricted here to a two-classes problem with various distances.

The results are brought in Tables 2 and 3. The nearest neighbor classifier works quite well for classification of this data set, making it somewhat difficult to see big differences in the performance of the different methods. Nonetheless, for this classifier the geometric methods dominate, with the log-Euclidean metric achieving the highest overall accuracy rate. Using the MDM classifier the classification rates are generally lower and it is more readily seen that gaPCA using the log-Euclidean metric has a clear advantage for this task.

5.3 Brain-computer interface

The use of covariance matrices is prevalent in the brain computer interface (BCI) community (Barachant et al. 2010; Lotte and Guan 2011). EEG signals involve highly complex and non-linear phenomenon (Blankertz et al. 2008) which cannot be modeled efficiently using simple Gaussian assumptions. In this context, for some specific applications, covariance matrices (using their natural Riemannian geometry) have been successfully used (Barachant et al. 2010, 2012, 2013; Yger 2013). As emphasized in Blankertz et al. (2008) and Lotte and Guan (2011), dimensionality reduction and spatial filtering are crucial steps for building an efficient BCI system. Hence, an unsupervised dimensionality reduction method utilizing the Riemannian geometry of covariance matrices is of great interest for BCI applications.

In this set of experiments, we apply our methods to BCI data from the *BCI competition III datasets IIIa and IV* (Schlögl et al. 2005). These datasets contain motor imagery (MI) EEG signals and was collected in a multi-class setting, with the subjects performing more than 2 different MI tasks. As was done in Lotte and Guan (2011), we evaluate our algorithms on two-class problems by selecting only signals of left- and right-hand MI trials.

Dataset IIIa comprises EEG signals recorded from 60 electrodes from three subjects who performed left-hand, right-hand, foot and tongue MI. A training set and a test set are available for each subject. Both sets contain 45 trials per class for Subject 1, and 30 trials per class for Subjects 2 and 3. Dataset IV, comprises EEG signals recorded from 118 electrodes from five subjects who performed left-hand, right-hand, foot and tongue MI. Here 280 trials were available for each subject, among which 168, 224, 84, 56 and 28 composed the training sets for the respective subjects. The remaining trials composed their test sets.

We apply the same pre-processing as described in Lotte and Guan (2011). EEG signals were band-pass filtered in 8-30 Hz, using a 5th order Butterworth filter. For each trial, we extracted features from the time segment located from 0.5 to 2.5 s after the cue instructing the subject to perform MI.

For both datasets we reduce the matrices from their original size to 6×6 as it corresponds to the number of sources recommended in Blankertz et al. (2008) for *common spatial pattern*

Table 2 Accuracy rates for the various PCA methods using a minimum distance to the mean (MDM) classifier

Textures	58 v 111	5 v 72	62 v 111	25 v 91	92 v 101	44 v 89	13 v 19	85 v 103	avg
No compression	81	69	82	88	78	41	52	82	71.625
2DPCA	81	69	82	88	78	41	52	82	71.625
PGA	93	67	84	94	**84**	65	53	79	77.375
δ_rPCA	89	74	**100**	99	75	**68**	62	83	81.25
δ_ePCA	81	69	82	88	78	41	52	82	71.625
δ_sPCA	89	68	**100**	92	83	**68**	53	81	79.25
δ_{le}PCA	**97**	**79**	91	**100**	71	63	**66**	**85**	**81.5**

In each column the best method is highlighted by boldface

Table 3 Accuracy rates for the various PCA methods using a nearest neighbor classifier

Textures	58 v 111	5 v 72	62 v 111	25 v 91	92 v 101	44 v 89	13 v 19	85 v 103	avg
No compression	99	79	99	99	**96**	44	72	73	82.625
2DPCA	99	79	99	99	**96**	44	72	73	82.625
PGA	92	72	72	**100**	90	**84**	67	81	82.25
δ_rPCA	96	81	**100**	99	91	66	**78**	74	85.625
δ_ePCA	99	79	99	99	**96**	44	72	73	82.625
δ_sPCA	96	81	**100**	95	92	63	**78**	74	84.875
δ_{le}PCA	**100**	**96**	**100**	**100**	93	68	75	**82**	**89.25**

In each column the best method is highlighted by boldface

Table 4 Accuracy rates for the various PCA methods using the Riemannian metric

Subject	Data set IIIa				Data set IV					
	1	2	3	Avg	1	2	3	4	5	Avg
No compression	**95.56**	60	**98.33**	**84.63**	53.57	76.79	53.06	49.11	69.05	60.32
2DPCA	84.44	60	73.33	72.59	54.46	71.43	53.57	**66.07**	58.33	60.77
δ_rPCA	**95.56**	**68.33**	85	82.96	**55.36**	**94.64**	52.04	50.89	68.65	64.32
δ_cPCA	**95.56**	**68.33**	85	82.96	**55.36**	**94.64**	52.04	50.89	68.65	64.32
δ_ePCA	84.44	60	73.33	72.59	**55.36**	73.21	53.57	65.62	58.33	61.22
δ_sPCA	**95.56**	61.67	85	80.74	**55.36**	64.29	52.04	54.02	53.97	55.94
δ_{le}PCA	86.67	61.67	85	77.78	53.57	76.79	50.51	52.23	51.19	56.86
δ_gPCA	62.22	50	50	54.07	46.43	50	50	50.89	48.41	49.15
PGA	76.67	50	78.33	68.33	54.46	75	**59.69**	64.29	**69.84**	**64.66**
CSP + LDA	95.56	61.67	93.33	83.52	66.07	96.43	47.45	71.88	49.6	66.29

The best method (excluding CSP+LDA) is highlighted by boldface

(CSP). Since our problem is non-convex, we restarted the optimization 5 times with different initial values and used the matrix W which produced the lowest value of the cost function. As for the image data, the performance measure for the dimensionality reduction was the classification accuracy with MDM classification. We used both the Riemannian and the Euclidean metrics to compute the class means and distances for the test samples. However, we report the results only for the Riemannian metric, as they were better for all subjects. The results using the Euclidean metric can be found in "Appendix 5".

The accuracy rates of the classification are presented in Table 4. As a reference on these datasets, we also report the results of a classical method of the literature. This method (Lotte and Guan 2011) consists of *supervised* dimensionality reduction, namely a CSP, followed by a linear discriminant analysis on the log-variance of the sources extracted by the CSP.

While the results of Lotte and Guan (2011) cannot be compared to those of our *unsupervised* techniques in a straightforward manner, they nonetheless serve as a motivation.[6] Since we intend to extend our approach to the supervised learning setting in our future work, it is instructive to quantitatively assess the performance gap even at this stage of research. Encouragingly, our comparatively naive methods work well, obtaining the same classification rates as Lotte and Guan (2011) for some test subjects, and for others even achieving higher rates.

Once again, our δ_cPCA method yields the same classification accuracy as δ_r, albeit for slightly different transformation matrices. Once again this implies that our approximation for the Riemannian mean of the compressed set $\mathbf{X}^{\downarrow}$ is adequate.

5.4 Approximate versus exact costs

Our experiments indicate that the use of the exact method δ_c is effectively equivalent to using the approximate method δ_r. So, to determine which method is preferable, we compare the time complexity of each optimization step. When initialized identically, the number of steps needed to reach convergence is roughly the same for both methods. In terms of

[6] For the same reason we cannot directly compare our results to the work of Harandi et al. (2014a) or Harandi and Salzmann (2015).

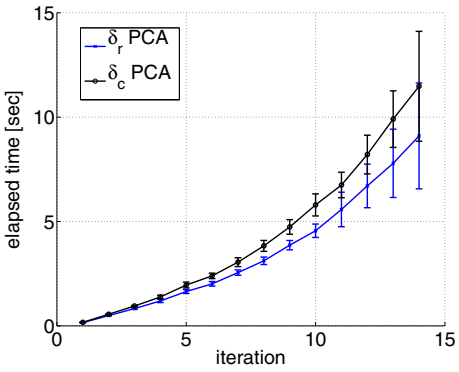

Fig. 7 Comparison between run time of δ_r and δ_c formulations

matrix inversions and decompositions, it can be seen from the expressions for the gradients that both methods require roughly the same amount of operations. However, the gradient computation for δ_c requires the multiplication of larger matrices, as well as the computation of the Riemannian mean of the set $\mathbf{X}^{\downarrow}$ for each optimization step. Note that this is the mean in the smaller *compressed* space, and so this does not significantly increase the run time.

Figure 7 shows the average time elapsed until completion of the iterate, in seconds, for both methods. We see that the approximate δ_r runs faster than the exact δ_c. Since the gain in performance is negligible, we conclude that the use of the formulation optimizing the approximate Fréchet variance is favorable.

5.5 Choosing a metric

Our experiments have shown that our geometric method gaPCA in general performs better than other existing PCA methods. In terms of choosing which metric to use, each offers a different appeal: The Euclidean geometry, while simple and efficient, is usually too simplistic to capture the curvature of the space. The AIRM and the closely related log-determinant metrics are the most natural and offer the most invariance properties. The log-Euclidean metric also exhibits many invariance properties and can be seen as a compromise between the simplicity of the Euclidean metric and the complexity of the AIRM. Thus, an adequate metric must be suited to the data at hand.

6 Conclusion

In this paper, we introduced a novel way to perform unsupervised dimensionality reduction for SPD matrices. We provided a rectified formulation of matrix PCA based on the optimization of a generalized notion of variance for SPD matrices. Extending this formulation to other geometries, we used tools from the field of optimization on manifolds. We showed that it suffices to use a simple approximation for the Fréchet variance and that it is not necessary to use a more complex formulation in order to optimize it precisely. We applied our method to synthetic and real-world data and demonstrated its usefulness.

In future work we consider several promising extensions to our methods. First, we may cast our δPCA to a stochastic optimization setting on manifolds (Bonnabel 2013). Such an approach may be useful for the massive datasets common in applications such as computer vision. In addition, it would be interesting to use our approach with criteria in the spirit of Yger

and Sugiyama (2015). This would lead to supervised dimensionality reduction, bridging the gap between the supervised log-Euclidean metric learning proposed in Yger and Sugiyama (2015) and the dimensionality reduction proposed in Harandi et al. (2014a).

Acknowledgements During this work, I.H. was supported by a MEXT Scholarship and F.Y. by a JSPS fellowship. The authors would like to acknowledge the support of KAKENHI 23120004 (for I.H.), KAKENHI 2604730 (for F.Y.) and KAKENHI 25700022 (for M.S.). F.Y. would like to thank Dr. M. Harandi for interesting and stimulating discussions.

Appendices

In this appendix, we assume X and Y to be two SPD matrices of size $n \times n$ and W a $n \times p$ matrix in a low-rank manifold.

Appendix 1: Cost function derivative for δ_{e}

In our matrix Euclidean PCA, the cost is much simpler to derive. In this method, we want to learn a full column rank matrix W by minimizing a cost function based on $f(W) = \delta_{\mathrm{e}}^2(W^\top X W, W^\top Y W)$ where δ_{e} is defined as

$$\delta_{\mathrm{le}}^2(X, Y) = \|X - Y\|_{\mathcal{F}}^2 .$$

As for the logEuclidean case, we reformulate the cost:

$$\begin{aligned} f(W) &= \left\| W^\top X W - W^\top Y W \right\|_{\mathcal{F}}^2 \\ &= \left\langle W^\top X W - W^\top Y W, W^\top X W - W^\top Y W \right\rangle \\ &= \left\langle W^\top X W, W^\top X W \right\rangle - 2\left\langle W^\top X W, W^\top Y W \right\rangle + \left\langle W^\top Y W, W^\top Y W \right\rangle . \end{aligned}$$

Then, reusing Eq. (21) for the directional derivative for the quadratic term $W^\top X W$ and making use of the composition and product rules (defined in the previous section), we have:

$$\begin{aligned} Df(W)[H] &= 2\left\langle H^\top X W + W^\top X H, W^\top X W \right\rangle + 2\left\langle H^\top Y W + W^\top Y H, W^\top Y W \right\rangle \quad (10) \\ &\quad - 2\left\langle H^\top X W + W^\top X H, W^\top Y W \right\rangle - 2\left\langle H^\top Y W + W^\top Y H, W^\top X W \right\rangle \\ &= 2\left\langle H^\top X W + W^\top X H, W^\top (X - Y) W \right\rangle \\ &\quad + 2\left\langle H^\top Y W + W^\top Y H, W^\top (Y - X) W \right\rangle \\ &= 4\left\langle H^\top X W, W^\top (X - Y) W \right\rangle - 4\left\langle H^\top Y W, W^\top (X - Y) W \right\rangle \\ &= 4\left\langle H^\top (X - Y) W, W^\top (X - Y) W \right\rangle \\ Df(W)[H] &= \left\langle 4 (X - Y) W W^\top (X - Y) W, H \right\rangle \quad (11) \end{aligned}$$

Hence, we have:

$$\nabla f(W) = 4 (X - Y) W W^\top (X - Y) W \quad (12)$$

Appendix 2: Cost function derivative for δ_r

In Harandi et al. (2014a), a cost function similar to Eq.(1) and corresponding gradient function are derived. Since that gradient formulation is more computationally efficient than ours, we used it in our implementation. However, for the sake of completeness we include below an alternative gradient formulation.

While the definition of our objective function is quite intuitive, computing its derivative w.r.t. W for the purpose of optimization is not straight forward. First, for ease of notation, we define $f(W) = \delta_r^2(W^\top XW, W^\top YW)$. We compute the gradient based on $Df(W)[H]$, the directional derivative of f at W in the direction H.

As the directional derivative of the function $X \mapsto X^{-1/2}$ is not obvious to obtain, let us reformulate $f(W)$:

$$\begin{aligned}
& f(W) \\
&= \operatorname{tr}\left(\log\left(\left(W^\top XW\right)^{-1/2} W^\top YW \left(W^\top XW\right)^{-1/2}\right)\right. \\
&\qquad \left.\times \log\left(\left(W^\top XW\right)^{-1/2} W^\top YW \left(W^\top XW\right)^{-1/2}\right)\right) \\
&= \operatorname{tr}\left(\log\left(\underbrace{\left(W^\top XW\right)^{-1} W^\top YW}_{g_{XY}(W)}\right) \log\left(\left(W^\top XW\right)^{-1} W^\top YW\right)\right) \\
&= \langle \log\left(g_{XY}\left(W\right)\right), \log\left(g_{XY}\left(W\right)\right)\rangle
\end{aligned}$$

Next, owing to the product rule and the chain rule of the Fréchet derivative (Absil et al. 2009), we express $Dg_{XY}(W)[H]$ as

$$\begin{aligned}
Dg_{XY}(W)[H] &= D\left(X \mapsto X^{-1}\right)\left(W^\top XW\right)[W^\top XH \\
&\quad + H^\top XW]W^\top YW + \left(W^\top XW\right)^{-1}\left(W^\top YH + H^\top YW\right) \\
&= -\left(W^\top XW\right)^{-1}\left(W^\top XH + H^\top XW\right)\left(W^\top XW\right)^{-1} W^\top YW \\
&\quad + \left(W^\top XW\right)^{-1}\left(W^\top YH + H^\top YW\right) \\
&= -\tilde{X}^{-1}\left(W^\top XH + H^\top XW\right)\tilde{X}^{-1}\tilde{Y} + \tilde{X}^{-1}\left(W^\top YH + H^\top YW\right),
\end{aligned}$$

where for simplicity we have introduced the notation $\tilde{X} = W^\top XW$ and similarly $\tilde{Y} = W^\top YW$. The function $g_{XY}(W)$ can then be written as $g_{XY}(W) = \tilde{X}^{-1}\tilde{Y}$.

Note that the matrix $\tilde{X}^{-1}\tilde{Y}$, while in general is not a symmetric matrix, has real, positive eigenvalues and is diagonizable (Boumal 2010, Prop. (5.3.2)) as $\tilde{X}^{-1}\tilde{Y} = V\Lambda V^{-1}$.

In order to compute $Df(W)[H]$, let us introduce $\tilde{H} = V^{-1}\left(Dg_{XY}(W)[H]\right)V$ and $\tilde{F}$, a matrix of the first divided differences (Bhatia 1997, pp. 60, 164) of the log function for $\lambda_i = \Lambda_{ii}$. The symbol $\odot$ denotes the Hadamard product of two matrices. Then we have

$$\begin{aligned}
Df(W)[H] &= 2\left\langle D\log \circ g_{XY}\left(W\right)[H], \log \circ g_{XY}\left(W\right)\right\rangle && (13) \\
&= 2\left\langle D\log\left(g_{XY}\left(W\right)\right)[Dg_{XY}(W)[H]], \log\left(\tilde{X}^{-1}\tilde{Y}\right)\right\rangle && (14) \\
&= 2\left\langle \tilde{H} \odot \tilde{F}, V^\top \log\left(\tilde{X}^{-1}\tilde{Y}\right) V^{-\top}\right\rangle && (15)
\end{aligned}$$

$$= 2\left\langle \underbrace{\left(V^\top \log\left(\tilde{X}^{-1}\tilde{Y}\right)V^{-\top}\right)\odot \tilde{F}}_{A}, \tilde{H}^\top \right\rangle \tag{16}$$

$$= 2\left\langle VAV^{-1}, Dg_{XY}(W)[H]^\top \right\rangle \tag{17}$$

$$= 2\left\langle \underbrace{VAV^{-1}\tilde{X}^{-1}}_{B}, W^\top YH + H^\top YW \right\rangle \tag{18}$$

$$-2\left\langle \underbrace{\tilde{X}^{-1}\tilde{Y}VAV^{-1}\tilde{X}^{-1}}_{C}, W^\top XH + H^\top XW \right\rangle$$

$$= \left\langle \underbrace{2YW\left(B+B^\top\right) - 2XW\left(C+C^\top\right)}_{\nabla f(W)}, H \right\rangle, \tag{19}$$

where the transition between Eqs. (15) and (16) is due to the identity $\langle A\odot B, C\rangle = \left\langle A\odot C^\top, B^\top\right\rangle$ (Boumal and Absil 2011, Eq. (5.5)).

Since the directional derivative $Df(W)[H]$ is related to its gradient by $Df(W)[H] = \langle \nabla f(W), H\rangle$, we have obtained the desired gradient:

$$\nabla f(W) = 2YW\left(B+B^\top\right) - 2XW\left(C+C^\top\right). \tag{20}$$

Appendix 3: Cost function derivative for δ_{l}

In our logEuclidean PCA, we want to learn a full column-rank matrix W by minimizing a cost function based on $f(W) = \delta^2_{\mathrm{le}}(W^\top XW, W^\top YW)$ where δ_{le} is defined as

$$\delta^2_{\mathrm{le}}(X,Y) = \|\log(X) - \log(Y)\|^2_{\mathcal{F}}.$$

Let us reformulate the directional derivative of f:

$$\begin{aligned} f(W) &= \left\|\log\left(W^\top XW\right) - \log\left(W^\top YW\right)\right\|^2_{\mathcal{F}} \\ &= \left\langle \log\left(W^\top XW\right) - \log\left(W^\top YW\right), \log\left(W^\top XW\right) - \log\left(W^\top YW\right)\right\rangle \\ &= \left\langle \log\left(W^\top XW\right), \log\left(W^\top XW\right)\right\rangle - 2\left\langle \log\left(W^\top XW\right), \log\left(W^\top YW\right)\right\rangle \\ &\quad + \left\langle \log\left(W^\top YW\right), \log\left(W^\top YW\right)\right\rangle. \end{aligned}$$

In order to obtain ∇f, the (Euclidean) gradient of f, we first express $Df(W)[H]$ the directional derivative of f (at W in the direction H). This is due to the fact that $Df(W)[H] = \langle \nabla f(W), H\rangle$.

We recall that the directional derivative is defined as

$$Df(X)[H] = \lim_{h\to 0} \frac{f(X+hH) - f(X)}{h}.$$

As summarized in Boumal (2010, p. 53), the directional derivative is equipped with various useful identities such as:

$$D\left(f \circ g\right)(X)[H] = Df\left(g\left(X\right)\right)\left[Dg\left(X\right)[H]\right] \qquad \text{(composition rule)}$$
$$D\left(X \mapsto \left\langle f\left(X\right), g\left(X\right)\right\rangle\right)(X)[H] = \left\langle Df\left(X\right)[H], g\left(X\right)\right\rangle + \left\langle f\left(X\right), Dg\left(X\right)[H]\right\rangle \qquad \text{(product rule)}.$$

Moreover, from the definition of the directional derivative, we can show that for a symmetric matrix A:

$$D\left(X \mapsto X^\top AX\right)(X)[H] = H^\top AX + X^\top AH \tag{21}$$

Now, using these identities we find the derivative of f:

$$\begin{aligned}
Df(W)[H] &= 2\left\langle D\log\left(W^\top XW\right)[H^\top XW + W^\top XH], \log\left(W^\top XW\right)\right\rangle && (22)\\
&\quad + 2\left\langle D\log\left(W^\top YW\right)[H^\top YW + W^\top YH], \log\left(W^\top YW\right)\right\rangle\\
&\quad - 2\left\langle D\log\left(W^\top XW\right)[H^\top XW + W^\top XH], \log\left(W^\top YW\right)\right\rangle\\
&\quad - 2\left\langle D\log\left(W^\top YW\right)[H^\top YW + W^\top YH], \log\left(W^\top XW\right)\right\rangle\\
&= 2\left\langle D\log\left(W^\top XW\right)[H^\top XW + W^\top XH], \log\left(W^\top XW\right) - \log\left(W^\top YW\right)\right\rangle && (23)\\
&\quad + 2\left\langle D\log\left(W^\top YW\right)[H^\top YW + W^\top YH], \log\left(W^\top YW\right) - \log\left(W^\top XW\right)\right\rangle\\
&= 2\left\langle D\log\left(W^\top XW\right)\left[\log\left(W^\top XW\right) - \log\left(W^\top YW\right)\right], H^\top XW + W^\top XH\right\rangle && (24)\\
&\quad + 2\left\langle D\log\left(W^\top YW\right)\left[\log\left(W^\top YW\right) - \log\left(W^\top XW\right)\right], H^\top YW + W^\top YH\right\rangle\\
Df(W)[H] &= \left\langle 4XWD\log\left(W^\top XW\right)\left[\log\left(W^\top XW\right) - \log\left(W^\top YW\right)\right], H\right\rangle\\
&\quad + \left\langle 4YWD\log\left(W^\top YW\right)\left[\log\left(W^\top YW\right) - \log\left(W^\top XW\right)\right], H\right\rangle && (25)
\end{aligned}$$

From the expression of the function f, we first apply the product rule and the chain rule in order to obtain Eq. 22. Then, from Eqs. 23 and 24, we use the property that $Dlog(X)[.]$ is an auto-adjoint operator[7] for symmetric definite positive matrices, as stated in Boumal and Absil (2011) and demonstrated in Boumal (2010, Chap. 5 p. 52).

Note that the directional derivative of the matrix logarithm can be computed numerically thanks to the algorithm provided in Boumal (2010) and Boumal and Absil (2011).

Hence, we have:

$$\begin{aligned}
\nabla f(W) &= 4XWD\log\left(W^\top XW\right)\left[\log\left(W^\top XW\right) - \log\left(W^\top YW\right)\right]\\
&\quad + 4YWD\log\left(W^\top YW\right)\left[\log\left(W^\top YW\right) - \log\left(W^\top XW\right)\right]
\end{aligned} \tag{26}$$

[7] This means that for all symmetric matrices H_1 and H_2, we have $\left\langle D\log\left(X\right)[H_1], H_2\right\rangle = \left\langle H_1, D\log\left(X\right)[H_2]\right\rangle$.

Appendix 4: Cost function derivative of the constrained cost function

We derive the Euclidean gradient of the following cost function:

$$\hat{f}(W) = f(W, \Lambda) = \sum_{j=1}^{n} \delta_{\mathrm{r}}^2 \left(W^\top X_j W, \Lambda \right) \text{ s.t.}$$

$$C(W, \Lambda) = \sum_{j=1}^{n} \log \left(\left(W^\top X_j W \right)^{-1/2} \Lambda \left(W^\top X_j W \right)^{-1/2} \right) = 0^{p \times p} \tag{27}$$

where $W \in \mathcal{S}(n, p)$, the set of all rank-p matrices of size $n \times p$, $\{X_j\} \subset \mathcal{S}_+^n$ and $\delta_{\mathrm{r}}^2(A, B) = \mathrm{tr}\left(\log^2\left(A^{-1/2} B A^{-1/2}\right)\right)$ is the AIRM. For brevity we leave out the dependence of f and C on the set $\{X_j\}$.

For a given matrix W, $\Lambda \in \mathcal{S}_+^p$ is the Riemannian mean of the set of compressed matrices $\{W^\top X_j W\}$. This is embodied by the constraint, which stems from the fact that the Riemannian mean uniquely minimizes the Fréchet variance $\sum_{j=1}^{n} \delta_{\mathrm{r}}^2 \left(W^\top X_j W, \Lambda \right)$.

To simplify the computation we use an equivalent expression for the AIRM, $\delta_{\mathrm{r}}^2(A, B) = \mathrm{tr}\left(\log^2\left(A^{-1} B\right)\right)$. The constraint then becomes

$$C(W, \Lambda) = \sum_{j=1}^{n} \log \left(\left(W^\top X_j W \right)^{-1} \Lambda \right) = 0^{p \times p}. \tag{28}$$

The gradient computation will be done using a vectorized form of Eq. (27), namely

$$\hat{f}\left(\vec{W}\right) = f\left(\vec{W}, \vec{\Lambda}\right) \text{ s.t. } C\left(\vec{W}, \vec{\Lambda}\right) = 0^{p^2}, \tag{29}$$

where $\vec{(\cdot)}$ is shorthand for vec $(\cdot)$, the column-wise matrix vectorization operator. The function $C(W, \Lambda)$, originally a function mapping $\mathbb{R}^{n \times p} \times \mathbb{R}^{p \times p} \to \mathbb{R}^{p \times p}$, becomes in its vectorized form a mapping $\mathbb{R}^{np} \times \mathbb{R}^{p^2} \to \mathbb{R}^{p^2}$.

To compute the gradient we use the implicit function theorem. In order to invoke the theorem, the following conditions must hold (Krantz and Parks 2012):

- For all $\vec{W} \in \mathbb{R}^{np}$ there exists a unique $\vec{\Lambda} \in \mathbb{R}^{p^2}$ such that $C\left(\vec{W}, \vec{\Lambda}\right) = 0$.
- There exists an open set $D \subset \mathbb{R}^{np} \times \mathbb{R}^{p^2}$ with $\left\{ \left(\vec{W}, \vec{\Lambda}\right) : \vec{W} \in \mathbb{R}^{p^2},\ C\left(\vec{W}, \vec{\Lambda}\right) = 0 \right\} \subset D$ such that f and C are twice differentiable on D.
- The inverse $C_{\vec{\Lambda}}\left(\vec{W}, \vec{\Lambda}\right)^{-1}$ exists for all $\left(\vec{W}, \vec{\Lambda}\right) \in \left\{ \left(\vec{W}, \vec{\Lambda}\right) : \vec{W} \in \mathbb{R}^{p^2},\ C\left(\vec{W}, \vec{\Lambda}\right) = 0 \right\}$.

Under these assumptions, the implicit function theorem guarantees the existence of a differentiable function $\vec{\Lambda} : \mathcal{W} \to \mathbb{R}^{p^2}$ which is defined by $C\left(\vec{W}, \vec{\Lambda}\right) = 0$ and whose derivative is obtained by differentiating $C\left(\vec{W}, \vec{\Lambda}\right) = 0$. This gives

$$\vec{\Lambda}_{\vec{W}}\left(\vec{W}\right) = -C_{\vec{\Lambda}}\left(\vec{W}, \vec{\Lambda}\right)^{-1} C_{\vec{W}}\left(\vec{W}, \vec{\Lambda}\right). \tag{30}$$

Using the results above, the gradient of $\hat{f}\left(\vec{W}\right)$ is

$$\nabla \hat{f}\left(\vec{W}\right) = \vec{\Lambda}_{\vec{W}}\left(\vec{W}\right)^\top \nabla_{\vec{\Lambda}} f\left(\vec{W}, \vec{\Lambda}\right) + \nabla_{\vec{W}} f\left(\vec{W}, \vec{\Lambda}\right). \tag{31}$$

The partial derivatives $\nabla_\Lambda f(W,\Lambda)$ and $\nabla_W f(W,\Lambda)$ have been previously computed and are given by

$$\nabla_\Lambda f(W,\Lambda) = 2\Lambda^{-1}\log\left(\left(W^\top X_j W\right)^{-1}\Lambda\right) \tag{32}$$

$$\nabla_W f(W,\Lambda) = 4X_j W\left(W^\top X_j W\right)^{-1}\log\left(\left(W^\top X_j W\right)\Lambda^{-1}\right). \tag{33}$$

Note that it is the fact that the matrix log is squared in the AIRM allows us to use $\log\left(A^{-1}B\right)$ or $\log\left(B^{-1}A\right)$ as convenient.

We are left with the task of computing $C_{\vec{\Lambda}}\left(\vec{W},\vec{\Lambda}\right)$ and $C_{\vec{W}}\left(\vec{W},\vec{\Lambda}\right)$. For these we will need an expression for $\mathrm{d}\,\mathrm{vec}(\log(X))$:

First, we use the Mercator series $\log(I+X) = -\sum_{k=1}^{\infty}\frac{(-1)^k}{k}X^k$. Then,

$$\begin{aligned}\mathrm{d}\left[\log(I+X)\right] &= -\sum_{k=1}^{\infty}\frac{(-1)^k}{k}\left(\mathrm{d}X^k\right)\\ &= -\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{j=0}^{m}\left(X^j(\mathrm{d}X)X^{m-j}\right)\end{aligned} \tag{34}$$

Vectorizing, we get

$$\begin{aligned}\mathrm{d}\left[\mathrm{vec}(\log(I+X))\right] &= -\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{j=0}^{m}\mathrm{vec}\left(X^j(\mathrm{d}X)X^{m-j}\right)\\ &= -\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{j=0}^{m}\left(X^{m-j}\otimes X^j\right)\mathrm{vec}(\mathrm{d}X),\end{aligned} \tag{35}$$

where we have used the identity $\mathrm{vec}(AXB) = \left(B^\top\otimes A\right)\mathrm{vec}(X)$.

Next, we use the eigenvalue decomposition $X = UDU^\top$ where U is a unitary matrix $UU^\top = U^\top U = I$ and D is a diagonal matrix containing the (strictly positive) eigenvalues λ_i of the SPD matrix X.

Using the identity $AC\otimes BD = (A\otimes B)(C\otimes D)$, we get

$$X^{m-j}\otimes X^j = \left(UD^{m-j}U^\top\right)\otimes\left(UD^jU^\top\right) = (U\otimes U)\left(D^{m-j}\otimes D^j\right)\left(U^\top\otimes U^\top\right) \tag{36}$$

We compute the expression $-\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{j=0}^{m}\left(D^{m-j}\otimes D^j\right)$. For the series to converge, we assume without loss of generality that $\lambda_i < 1$. For $\lambda_i = \lambda_j = \lambda$ we have

$$-\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{j=0}^{m}\lambda^{m-j}\lambda^j = -\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{j=0}^{m}\lambda^m = -\sum_{m=0}^{\infty}(-1)^{m+1}\lambda^m = \frac{1}{1+\lambda} \tag{37}$$

For $\lambda_i \neq \lambda_j$ we have

$$\begin{aligned}-\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{k=0}^{m}\lambda_i^{m-k}\lambda_j^k &= -\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\sum_{k=0}^{m}\lambda_i^m\left(\frac{\lambda_j}{\lambda_i}\right)^k\\ &= -\sum_{m=0}^{\infty}\frac{(-1)^{m+1}}{m+1}\lambda_i^m\left(\frac{1-\left(\frac{\lambda_j}{\lambda_i}\right)^{m+1}}{1-\frac{\lambda_j}{\lambda_i}}\right)\end{aligned}$$

$$= -\sum_{m=0}^{\infty} \frac{(-1)^{m+1}}{m+1} \left(\frac{\lambda_i^{m+1} - \lambda_j^{m+1}}{\lambda_i - \lambda_j} \right)$$
$$= \frac{\log(1+\lambda_i) - \log(1+\lambda_j)}{(1+\lambda_i) - (1+\lambda_j)} \tag{38}$$

Putting this all together yields

$$\mathrm{d}\left[\mathrm{vec}\left(\log(X)\right)\right] = (U \otimes U)\, \tilde{D} \left(U^\top \otimes U^\top\right), \tag{39}$$

where $\tilde{D}$ is a diagonal matrix of size $p^2 \times p^2$ whose elements are given by:

$$D_{kk} = \phi(\lambda_i, \lambda_j) = \begin{cases} \frac{1}{\lambda_i} & \lambda_i = \lambda_j \\ \frac{\log(\lambda_i) - \log(\lambda_j)}{(\lambda_i) - (\lambda_j)} & \lambda_i \neq \lambda_j \end{cases} \tag{40}$$

and $i = \lfloor k/p \rfloor$ and $j = \mod(k, p)$.

In addition, we will need the following expressions:

$$\mathrm{d}\,\mathrm{vec}\left(A^{-1}\right) = -\left(A^{-1} \otimes A^{-1}\right) \mathrm{d}\,\mathrm{vec}(A) \tag{41}$$
$$\mathrm{d}\,\mathrm{vec}(AB) = \left(B^\top \otimes I\right) \mathrm{d}\,\mathrm{vec}(A) \tag{42}$$

and

$$\begin{aligned} \mathrm{d}\,\mathrm{vec}\left(A^\top B A\right) &= \mathrm{vec}\left[\mathrm{d}\left(A^\top\right) BA + A^\top B \mathrm{d}(A)\right] \\ &= \left(A^\top B \otimes I_p\right) \mathrm{d}\,\mathrm{vec}\left(A^\top\right) + \left(I_p \otimes A^\top B\right) \mathrm{d}\,\mathrm{vec}(A) \\ &= \left(I_{p^2} + K\right)\left(I_p \otimes A^\top B\right) \mathrm{d}\,\mathrm{vec}(A), \end{aligned} \tag{43}$$

where K is the *commutator* matrix (Abadir and Magnus 2005) that allows us to go from $\mathrm{d}\,\mathrm{vec}\left(A^\top\right)$ to $\mathrm{d}\,\mathrm{vec}(A)$ and to change the order of the arguments of the Kronecker product.

Equipped with the above and using the chain rule, our partial derivatives are:

$$\begin{aligned} C_{\vec{\Lambda}}\left(\vec{W}, \vec{\Lambda}\right) &= \sum_{j=1}^{n} \mathrm{d}_{\vec{\Lambda}}\, \mathrm{vec}\left(\log\left(\left(W^\top X_j W\right)^{-1} \Lambda\right)\right) \\ &= \sum_{j=1}^{n} \mathrm{d}\left[\mathrm{vec}\left(\log\left(\left(W^\top X_j W\right)^{-1} \Lambda\right)\right)\right]\left(\left(W^\top X_j W\right)^{-1} \otimes I_p\right) \end{aligned} \tag{44}$$

and

$$\begin{aligned} C_{\vec{W}}\left(\vec{W}, \vec{\Lambda}\right) &= \sum_{j=1}^{n} \mathrm{d}_{\vec{W}}\, \mathrm{vec}\left(\log\left(\left(W^\top X_j W\right)^{-1} \Lambda\right)\right) \\ &= -\sum_{j=1}^{n} \mathrm{d}\left[\mathrm{vec}\left(\log\left(\left(W^\top X_j W\right)^{-1} \Lambda\right)\right)\right](\Lambda \otimes I_p) \\ &\quad \left(\left(W^\top X_j W\right)^{-1} \otimes \left(W^\top X_j W\right)^{-1}\right)\left(I_{p^2} + K\right)\left(I_p \otimes W^\top X_j\right) \end{aligned} \tag{45}$$

Appendix 5: BCI classification results using Euclidean metric

Table 5 contains the results of BCI data classification using the MDM classifier with the Euclidean metric.

Table 5 Accuracy rates for the various PCA methods using the Euclidean metric

Subject	Data set IIIa				Data set IV					
	1	2	3	Avg	1	2	3	4	5	Avg
No compression	63.33	48.33	55	55.55	47.32	69.64	**54.59**	62.05	41.27	54.97
2DPCA	61.11	48.33	55	54.81	47.32	66.07	**54.59**	**62.5**	41.27	54.35
δ_rPCA	**86.67**	**61.67**	73.33	**73.89**	50	**83.93**	52.04	56.25	**76.59**	**63.76**
δ_cPCA	**86.67**	**61.67**	73.33	**73.89**	50	**83.93**	52.04	56.25	**76.59**	**63.76**
δ_ePCA	61.11	48.33	55	54.81	47.32	64.29	**54.59**	**62.5**	41.67	54.07
δ_sPCA	**86.67**	60	71.67	72.78	50.89	64.29	48.47	55.36	52.38	54.28
δ_{le}PCA	81.11	56.67	**80**	72.59	**52.68**	78.57	50.51	52.68	50.4	56.97
δ_gPCA	61.11	50	65	58.7	46.43	50	50.51	50.89	51.98	49.96
PGA	66.67	48.33	56.67	57.22	47.32	58.93	50	**62.5**	50.4	53.83
CSP + LDA	95.56	61.67	93.33	83.52	66.07	96.43	47.45	71.88	49.6	66.29

Best results are marked in boldface

References

Abadir, K. M., & Magnus, J. R. (2005). *Matrix algebra* (Vol. 1). Cambridge: Cambridge University Press.

Absil, P. A., Mahony, R., & Sepulchre, R. (2009). *Optimization algorithms on matrix manifolds*. Princeton: Princeton University Press.

Al-Mohy, A. H., & Higham, N. J. (2009). Computing the Fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM Journal on Matrix Analysis and Applications*, *30*(4), 1639–1657.

Anderson, T. W. (1963). Asymptotic theory for principal component analysis. *The Annals of Mathematical Statistics*, *34*(1), 122–148.

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

Arsigny, V., Fillard, P., Pennec, X., & Ayache, N. (2007). Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, *29*(1), 328–347.

Barachant, A., & Congedo, M. (2014). A plug and play P300 BCI using information geometry. arXiv preprint arXiv:1409.0107.

Barachant, A., Bonnet, S., Congedo, M., Jutten, C. (2010). Riemannian geometry applied to BCI classification. In *Latent variable analysis and signal separation*, pp. 629–636.

Barachant, A., Bonnet, S., Congedo, M., & Jutten, C. (2012). Multiclass brain-computer interface classification by Riemannian geometry. *IEEE Transactions on Biomedical Engineering*, *59*(4), 920–928.

Barachant, A., Bonnet, S., Congedo, M., & Jutten, C. (2013). Classification of covariance matrices using a Riemannian-based kernel for BCI applications. *Neurocomputing*, *112*, 172–178.

Bhatia, R. (1997). *Matrix analysis*. Berlin: Springer.

Bhatia, R. (2009). *Positive definite matrices*. Princeton: Princeton University Press.

Bhatia, R. (2013). The riemannian mean of positive matrices. In F. Nielsen & R. Bhatia (Eds.), *Matrix information geometry* (pp. 35–51). Springer, Berlin.

Bishop, C. (2007). *Pattern recognition and machine learning* (1st ed.). Information science and statistics. Springer-Verlag New York.
Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., & Müller, K. R. (2008). Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, *25*(1), 41–56.
Bonnabel, S. (2013). Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, *58*(9), 2217–2229.
Boumal, N. (2010) Discrete curve fitting on manifolds. Master's thesis, Université Catholique de Louvain.
Boumal, N., Absil, P.A. (2011) Discrete regression methods on the cone of positive-definite matrices. In *IEEE international conference on acoustics, speech and signal processing*, pp. 4232–4235.
Boumal, N., Mishra, B., Absil, PA., Sepulchre, R. (2014). Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research 15*:1455–1459. http://www.manopt.org.
Bridson, M. R., & Haefliger, A. (2011). *Metric spaces of non-positive curvature* (Vol. 319). Berlin: Springer.
Brodatz, P. (1966). *Textures: A photographic album for artists and designers*. New York: Dover.
Cherian, A., & Sra, S. (2014). Riemannian sparse coding for positive definite matrices. In *European conference on computer vision*, pp. 299–314.
Cherian, A., Sra, S., Banerjee, A., & Papanikolopoulos, N. (2011). Efficient similarity search for covariance matrices via the Jensen-Bregman LogDet divergence. In *IEEE international conference on computer vision (ICCV), IEEE*, pp. 2399–2406.
Cichocki, A., Cruces, S., & Amari, S.I. (2014). Log-determinant divergences revisited: Alpha–beta and gamma log-det divergences. arXiv preprint arXiv:1412.7146.
Dryden, I.L., Koloydenko, A., & Zhou, D. (2009). Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *The Annals of Applied Statistics*, *3*(3), 1102–1123.
Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, *20*(2), 303–353.
Fletcher, P. T., Lu, C., Pizer, S. M., & Joshi, S. (2004). Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, *23*(8), 995–1005.
Fréchet, M. (1948). Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l'institut Henri Poincaré, Presses Universitaires de France*, *10*, 215–310.
Gabor, D. (1946). Theory of communication. *Journal of the Institute of Electrical Engineers Part III*, *93*, 429–457.
Goh, A., & Vidal, R. (2008). Clustering and dimensionality reduction on Riemannian manifolds. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 1–7.
Harandi, M., & Salzmann, M. (2015). Riemannian coding and dictionary learning: Kernels to the rescue. In *IEEE conference on computer vision and pattern recognition (CVPR)*.
Harandi, M., Sanderson, C., Wiliem, A., & Lovell, B.C. (2012). Kernel analysis over Riemannian manifolds for visual recognition of actions, pedestrians and textures. In *IEEE workshop on applications of computer vision (WACV)*, pp. 433–439.
Harandi, M., Salzmann, M., & Hartley, R. (2014a). From manifold to manifold: geometry-aware dimensionality reduction for SPD matrices. In *European conference on computer vision*, pp. 17–32.
Harandi, M., Salzmann, M., Porikli, F. (2014b). Bregman divergences for infinite dimensional covariance matrices. In *IEEE conference on computer vision and pattern recognition (CVPR)*.
Ho, J., Xie, Y., & Vemuri, B. (2013). On a nonlinear generalization of sparse coding and dictionary learning. In *International conference on machine learning*, pp. 1480–1488.
Huckemann, S., Hotz, T., & Munk, A. (2010). Intrinsic shape analysis: Geodesic PCA for Riemannian manifolds modulo isometric Lie group actions. *Statistica Sinica*, *20*, 1–100.
Jayasumana, S., Hartley, R., Salzmann, M., Li, H., & Harandi, M.(2013). Kernel methods on the Riemannian manifold of symmetric positive definite matrices. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 73–80.
Jolliffe, I. (2002). *Principal component analysis*. Berlin: Springer.
Krantz, S. G., & Parks, H. R. (2012). *The implicit function theorem: History, theory, and applications*. Berlin: Springer.
Kusner, M.J., Kolkin, N.I., Tyree, S., & Weinberger, K.Q. (2014). Stochastic covariance compression. arXiv preprint arXiv:1412.1740.
Lotte, F., & Guan, C. (2011). Regularizing common spatial patterns to improve BCI designs: Unified theory and new algorithms. *IEEE Transactions on Biomedical Engineering*, *58*(2), 355–362.
Lu, H., Plataniotis, K. N., & Venetsanopoulos, A. N. (2006). Multilinear principal component analysis of tensor objects for recognition. *International Conference on Pattern Recognition*, *2*, 776–779.
Mestre, X. (2008). Improved estimation of eigenvalues and eigenvectors of covariance matrices using their sample estimates. *IEEE Transactions on Information Theory*, *54*(11), 5113–5129.

Pennec, X., Fillard, P., & Ayache, N. (2006). A Riemannian framework for tensor computing. *International Journal of Computer Vision*, *66*(1), 41–66.

Schlögl, A., Lee, F., Bischof, H., & Pfurtscheller, G. (2005). Characterization of four-class motor imagery EEG data for the BCI-competition 2005. *Journal of Neural Engineering*, *2*(4), L14.

Sommer, S., Lauze, F., Hauberg, S., & Nielsen, M. (2010). Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations. In *European conference on computer vision*, pp. 43–56.

Sra, S. (2011). Positive definite matrices and the s-divergence. arXiv preprint arXiv:1110.1773.

Sra, S. (2012). A new metric on the manifold of kernel matrices with application to matrix geometric means. In *Advances in neural information processing systems* (Vol. 25, pp. 144–152).

Tuzel, O., Porikli, F., & Meer, P. (2006). Region covariance: A fast descriptor for detection and classification. In *European conference on computer vision*, pp. 589–600.

Tuzel, O., Porikli, F., & Meer, P. (2008). Pedestrian detection via classification on Riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(10), 1713–1727.

Yang, J., Zhang, D., Frangi, A. F., & Jy, Yang. (2004). Two-dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(1), 131–137.

Yger, F. (2013) A review of kernels on covariance matrices for BCI applications. In *IEEE international workshop on machine learning for signal processing*, pp. 1–6.

Yger, F., Sugiyama, M. (2015). Supervised log Euclidean metric learning for symmetric positive definite matrices. preprint arXiv:1502.03505.