JMLR: Workshop and Conference Proceedings 30:1–20, 2015 ACML 2015

Geometry-Aware Principal Component Analysis for Symmetric Positive Definite Matrices

Inbal Horev INBAL@MS.K.U-TOKYO.AC.JP
Tokyo Institute of Technology,
Graduate School of Information Science and Engineering, Department of Computer Science

Florian Yger FLORIAN@MS.K.U-TOKYO.AC.JP
Masashi Sugiyama SUGI@K.U-TOKYO.AC.JP
University of Tokyo,
Graduate School of Frontier Sciences, Department of Complexity Science and Engineering

Abstract

Symmetric positive definite (SPD) matrices, e.g. covariance matrices, are ubiquitous in machine learning applications. However, because their size grows as n^2 (where n is the number of variables) their high-dimensionality is a crucial point when working with them. Thus, it is often useful to apply to them dimensionality reduction techniques. Principal component analysis (PCA) is a canonical tool for dimensionality reduction, which for vector data reduces the dimension of the input data while maximizing 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 i) preserves more data variance by appropriately extending PCA to matrix data, and ii) 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.

Keywords: dimensionality reduction, PCA, Riemannian geometry, SPD manifold, Grassmann manifold

1. Introduction

Covariance matrices are used in a variety of machine learning applications. Three prominent examples are computer vision applications (Tuzel et al., 2008, 2006), brain imaging (Pennec et al., 2006; Arsigny et al., 2006) 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.

1.1. Geometry of covariance matrices

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

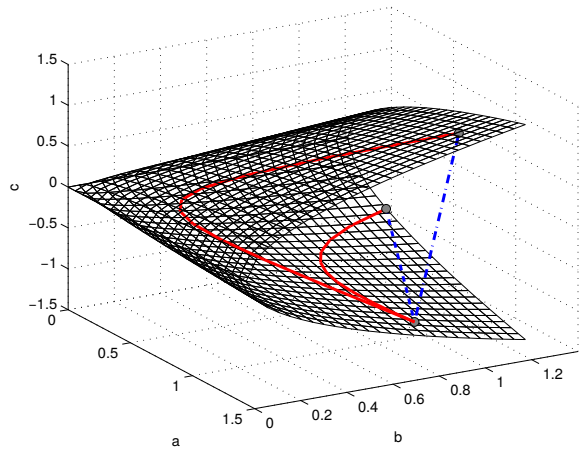
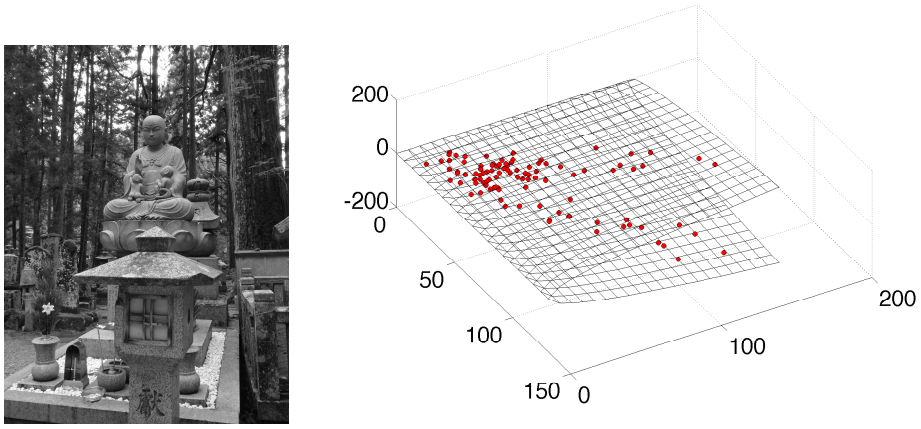


Figure 1: Comparison between Euclidean (blue straight dashed lines) and Riemannian (red curved solid lines) distances measured between points of the space $\mathcal{S}_+^2$.

straightforward approach for measuring similarity between SPD matrices would be to simply use the Euclidean distance derived from the Euclidean norm. This is readily seen in the following example for 2×2 SPD matrices. A matrix $A \in \mathcal{S}_+^2$ can be written as $A = \begin{bmatrix} a & c \\ c & b \end{bmatrix}$ 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 a convex cone which SPD matrices lie strictly within (see Fig. 1). In this representation, the Euclidean geometry of symmetric matrices then implies that distances are computed along straight lines (again, see Fig. 1).

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 already 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 distribution of these points exhibits some structure and the interior of the cone is not uniformly populated.

Despite its simplicity, the Euclidean geometry has several drawbacks and is not always well suited for SPD matrices (Fletcher et al., 2004; Arsigny et al., 2007; Sommer et al., 2010). For example, for a task as simple as averaging two matrices, it may occur that the determinant of the average is larger than any of the two matrices. This effect is an artifact of the Euclidean geometry and is referred to as the *swelling effect* by Arsigny et al. (2007). It is particularly harmful for data analysis as it adds spurious variation to the data. As another 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 (Dempster,



(a) (b)

Figure 2: Original image -courtesy of A. Bernard-Brunel- (left) and 2×2 covariance matrices (between the gradients of the image) 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.

1972). 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. Hence, in this Euclidean space interpolation between SPD matrices is possible, but extrapolation may produce indefinite matrices, leading to uninterpretable solutions.

In order to address these issues, an efficient alternative is to consider the space of SPD matrices as a curved space, namely a Riemannian manifold. For example, one of the possible Riemannian distances is computed on curved lines as illustrated in Fig. 1 for the space $\mathcal{S}_+^2$.

As noted in Fletcher et al. (2004) and Sommer et al. (2010), the use of the Riemannian geometry in a method is more natural as it ensures that the solutions will respect the constraint encoded by the manifold. In accordance with this approach, recently tools such as kernels (Barachant et al., 2013) and divergences (Sra, 2011; Cichocki et al., 2014), as well as methods such as dictionary learning (Ho et al., 2013; Cherian and Sra, 2014), metric learning (Yger and Sugiyama, 2015) and dimensionality reduction (Fletcher et al., 2004) have all been extended for SPD matrices using the Riemannian geometry.

1.2. Dimensionality reduction on manifolds

As discussed in Fletcher et al. (2004) and Harandi et al. (2014), dimensionality is a crucial point when working with covariance matrices. This is because their size grows as n^2 where n is the number of variables. Hence, 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 Section 2.2, while vector PCA is optimal in terms of preserving data variance, the commonly used naive extensions of vec-

tor 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 claim that a Euclidean formulation of PCA completely disregards the geometric structure of this space. If the *swelling effect*, stemming from the Euclidean geometry, distorts the results for simple tasks such as regression or averaging, it would certainly make it difficult to identify relevant information contained in a given dataset of SPD matrices. Thus, it would be disadvantageous to retain the principal modes of variations using the Euclidean geometry. A natural approach to cope with this issue is then to consider a Riemannian formulation of PCA which utilizes the intrinsic geometry of the SPD manifold.

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 correctly 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 for matrix data. As a result it captures more of the data variance. 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. More specifically, using the same transformation as in Harandi et al. (2014), we derive an unsupervised dimensionality reduction method maximizing a generalized variance of the data on the manifold. Through experiments on synthetic data and a signal processing application, we demonstrate the efficacy of our proposed dimensionality reduction method.

2. Geometry-aware 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, its naive extensions to matrices do not do so optimally for SPD matrices. In addition, 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 [1]. Other methods for dimensionality reduction of SPD matrices, while utilizing the structure of the SPD manifold, suffer from many of the same faults. First, they

1. 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. However, this is also the case for other geometries.

use only a local approximation of the manifold, causing a degradation in performance when distances are large. Second, and more importantly, these methods apply the same flawed formulation of matrix PCA.

We begin this section by stating a formal definition of our problem. Next, we show how to suitably extend PCA from the vector case to the matrix case so as to retain more of the data variance. Finally, we present several formulations for our geometry-aware methods and discuss some of their main properties. The proposed methods preserve more variance even when, as in the standard PCA, the Euclidean geometry is used.

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 $n \times n$ symmetric positive definite (SPD) matrices, and let $\mathbf{X} = \{X_i \in \mathcal{S}_+^n\}_{i=1}^N$ be a set of N instances in $\mathcal{S}_+^n$. Covariance matrices, widely used in many machine learning applications, are examples of SPD matrices. 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 thus 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. 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:

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

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

Definition 2 (Fréchet Variance) *The (sample) Fréchet variance of the set* $\mathbf{X}$ *w.r.t.* 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. (2014), 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 $W^\top W = \mathbb{I}_p$, the $p \times p$ identity matrix. The mapping then takes the form of $X^\downarrow = W^\top X W$.

2.2. Variance maximizing PCA for SPD matrices

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 (δ PCA) *δPCA 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 $\mathbb{R}^n$.

As described in Edelman et al. (1998) and Absil et al. (2009), such an optimization problem can be reformulated and efficiently solved on either the Stiefel or Grassmann manifolds. It would be straightforward to reformulate our problem on the Stiefel manifold, implying only an orthogonality constraint on W. However, as explained below (and also noted in Harandi et al. (2014)), our cost function is invariant under certain transformations. Since only the Grassmann manifold takes this invariance into account, we will consider a mapping $X^\downarrow = W^\top X W$ with $W \in \mathcal{G}(n,p)$.

Given our variance-based definition, it is only befitting that we compare it to ordinary PCA, the canonical method for dimensionality reduction which itself aims to preserve maximal data variance. For vector data, PCA is formulated as

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

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 main difference between our proposed method and ordinary PCA is that in our cost function, the matrix W acts on $\mathbf{X}$ on both sides. Although our method can accommodate multiple geometries via various choices of the metric δ, the difference between Eq. (3) and Eq. (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

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

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

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

In fact, the two expressions are in general not equivalent. Moreover, our proposed formulation consistently retains more of the data variance than the standard PCA method, as shown in Fig. 3. It is worth noting, however, that the two methods are equivalent when the matrices $\{X_i\}$ are jointly diagonalizable. In this case, the problem can be written in terms of the common basis. Then, the extra term $WW^\top$ does not contribute to the cost function and both methods yield identical results.

In order to make this article self-contained, we provide the Euclidean gradient of this cost function w.r.t. W (which we later use for the optimization):

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

Its derivation is detailed in Appendix C of the supplementary material.

2.3. Instantiation with different metrics

We have thus far addressed the issue of variance maximization in the Euclidean geometry, demonstrating that our proposed method results in improved retention of data variance (see Fig. 3). We next turn to the issue of geometry awareness.

As mentioned in the introduction, SPD matrices form a Riemannian manifold. So, it is natural to use a Riemannian metric rather than the Euclidean metric to measure the distance between two matrices. There are several choices of Riemannian metrics defined on the SPD manifold $\mathcal{S}_+^n$. A standard choice, due to its favorable geometric properties, is the affine invariant Riemannian metric (AIRM) (Bhatia, 2009).

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

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

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

Equipped with this metric, the SPD manifold becomes a complete manifold, i.e., all geodesics are contained within the manifold. This prevents the swelling effect and allows for matrix extrapolation without obtaining non-definite matrices. To see this, note that an extrapolated matrix lays on the extension of the geodesic between two matrices. For a complete manifold it is necessarily a valid element of the manifold. However, for a none complete manifold it may escape the boundaries of the manifold. We refer the reader to Figure 1. In addition, this metric introduces several invariance properties which we discuss in Section 2.4.

Using the AIRM, the cost function in Definition 3 then becomes

$$W = \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2\left(W^\top X_i W, W^\top \bar{X}_{\mathrm{r}} W\right) \tag{5}$$

with $\bar{X}_{\mathrm{r}}$ the Fréchet mean w.r.t. the AIRM δ_{r}.

Once again, we provide the Euclidean gradient of the cost function w.r.t. W (following Harandi et al. (2014))[3]:

$$D_W \delta_{\mathrm{r}}^2 \left(W^\top X_i W, W^\top \bar{X}_{\mathrm{r}} W \right) = \\ 4 \left(X_i W \left(W^\top X W \right)^{-1} - \bar{X}_{\mathrm{r}} W \left(W^\top \bar{X}_{\mathrm{r}} W \right)^{-1} \right) \log \left(W^\top X_i W \left(W^\top \bar{X}_{\mathrm{r}} W \right)^{-1} \right). \quad (6)$$

In general, the operations of computing the mean and projecting onto the lower dimensional manifold are not interchangeable. That is, let $\bar{X}^{\downarrow}_{\mathrm{r}}$ be the mean of the compressed set $\mathbf{X}^{\downarrow} = \left\{ W^\top X_i W \right\}$ and let $W^\top \bar{X}_{\mathrm{r}} W$ be the compressed mean of the original set $\mathbf{X}$. Then the two are not equal in general. Since we do not know in advance the mean of the compressed set, the cost function defined in Eq. 5 does not exactly express the Fréchet variance of $\mathbf{X}^{\downarrow}$. Rather, it serves as an approximation to it.

In an attempt to address this issue we may also consider a two-step mini-max formulation. In this formulation we alternate between i) optimization on W and ii) computation of the mean of the compressed set using the newly optimized W:

$$\begin{aligned} W_{k+1} &= \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2 \left(W^\top X_i W, \bar{X}_k \right), \\ \bar{X}_{k+1} &= \underset{X \in \mathcal{S}_+^p}{\operatorname{argmin}} \sum_i \delta_{\mathrm{r}}^2 \left(W_{k+1}^\top X_i W_{k+1}, X \right). \end{aligned} \quad (7)$$

Unfortunately, in our preliminary experiments, we were unable to obtain a stable solution using method (7). Investigation of the mini-max formulation is left as a topic for future work.

Instead, we study a variation of our intrinsic formulation whereby before optimizing over W, we first center the data. 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 $\mathbb{I}_n$. We call this method δ_{g}PCA, for reasons explained below. Its cost function is given by

$$W = \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2 \left(W^\top \tilde{X}_i W, \mathbb{I}_p \right). \quad (8)$$

It is interesting to examine the relation between the two cost functions in Eq. (5) and Eq. (8). Beginning with our definition of δ_{g}PCA, we write

$$\begin{aligned} & \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2 \left(W^\top \tilde{X}_i W, \mathbb{I}_p \right) \\ = & \underset{W \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2 \left(W^\top \bar{X}_{\mathrm{r}}^{-1/2} X_i \bar{X}_{\mathrm{r}}^{-1/2} W, W^\top \bar{X}_{\mathrm{r}}^{-1/2} \bar{X}_{\mathrm{r}} \bar{X}_{\mathrm{r}}^{-1/2} W \right) \\ = & \underset{\tilde{W}^\top \bar{X}_{\mathrm{r}}^{1/2} \tilde{W} \in \mathcal{G}(n,p)}{\operatorname{argmax}} \sum_i \delta_{\mathrm{r}}^2 \left(\tilde{W}^\top X_i \tilde{W}, \tilde{W}^\top \bar{X} \tilde{W} \right), \end{aligned} \quad (9)$$

3. 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 A of the supplementary material. In our experiments, as it was computationally more efficient, we use Eq. (6) for the gradient.

where $\tilde{W} = \bar{X}_{\mathrm{r}}^{-1/2} W$.

Comparing Eq. (5) to Eq. (9), we see that the two have the same form. However, the solution $\tilde{W}$ does not belong to the Grassmann manifold, but rather to a generalized Grassmann manifold, weighted by $\bar{X}_{\mathrm{r}}$ (hence the name 'g' of the method). In other words, we have $\tilde{W}^\top \bar{X}_{\mathrm{r}} \tilde{W} = \mathbb{I}_p$ instead of the standard $W^\top W = \mathbb{I}_p$.

In addition to the generalized Grassmann variation, we study two more variants of our δPCA. These are based on other metrics defined on the SPD manifold, namely the log-Euclidean metric (Arsigny et al., 2007) and the symmetrized log-determinant divergence (also referred to as the symmetric Stein loss) (Sra, 2012, 2011).

First is the log-Euclidean metric which, like the AIRM metric, is a Riemannian metric. As illustrated in Yger and Sugiyama (2015), this metric uses the logarithmic map to project the matrices to the tangent space at the identity $\mathcal{T}_{\mathbb{I}}\mathcal{S}_+^n$, where the standard Euclidean norm is then used to measure distances between matrices:

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

$$\delta_{\mathrm{le}}^2(X, Y) = \|\log(X) - \log(Y)\|_{\mathcal{F}}^2 \tag{10}$$

The Euclidean gradient w.r.t. W of this cost function is given by

$$\begin{aligned} &D_W \delta_{\mathrm{le}}^2(W^\top X_i W, W^\top \bar{X}_{\mathrm{le}} W) = \\ &4\left(X_i W D \log\left(W^\top X_i W\right)\left[\log\left(W^\top X_i W\right) - \log\left(W^\top \bar{X}_{\mathrm{le}} W\right)\right]\right. \\ &\left. + \bar{X}_{\mathrm{le}} W D \log\left(W^\top \bar{X}_{\mathrm{le}} W\right)\left[\log\left(W^\top \bar{X}_{\mathrm{le}} W\right) - \log\left(W^\top X_i W\right)\right]\right), \end{aligned} \tag{11}$$

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. log-Euclidean metric. Note that there is no closed-form solution for $D\log(W)[H]$ but it can be computed efficiently (Boumal, 2010; Boumal and Absil, 2011; Al-Mohy and Higham, 2009). This derivation is given in Appendix B of the supplementary material.

Next is the log-determinant (symmetric Stein) metric (Sra, 2011) defined as follows:

Definition 6 (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*

$$\delta_{\mathrm{s}}^2(X, Y) = \log\left(\det\left((X+Y)/2\right)\right) - \log\left(\det(XY)\right)/2. \tag{12}$$

Although it is not a Riemannian metric, it approximates the AIRM and shares several of its geometric properties. So, for computational ease we use the Riemannian mean $\bar{X}_{\mathrm{r}}$ instead of the mean w.r.t. symmetric Stein metric $\bar{X}_{\mathrm{s}}$. For further discussion on the Stein metric and its differences from and similarities to the AIRM metric, we refer the readers to Sra (2011).

Owing to Harandi et al. (2014), the gradient w.r.t. W of the Stein metric based cost function is given by

$$\begin{aligned} D_W \delta_{\mathrm{s}}^2\left(W^\top X_i W, W^\top \bar{X}_{\mathrm{r}} W\right) &= \left(X_i + \bar{X}_{\mathrm{r}}\right) W \left(W^\top \frac{X_i + \bar{X}_{\mathrm{r}}}{2} W\right)^{-1} \\ &- X_i W \left(W^\top X_i W\right)^{-1} - \bar{X}_{\mathrm{r}} W \left(W^\top \bar{X}_{\mathrm{r}} W\right)^{-1}. \end{aligned} \tag{13}$$

2.4. Invariance properties of the intrinsic distances

As discussed in Bhatia (2009), δ_{r} is invariant[4] by the congruence transformation, meaning that for any matrix $V \in GL(n)$, the group of $n \times n$ invertible matrices, we have

$$\delta_{\mathrm{r}}(X, Y) = \delta_{\mathrm{r}}(V^\top XV, V^\top YV).$$

Note that this property is also shared by δ_{s} (Sra, 2012). Hence, the points established below for the Riemannian distance will also hold for the Stein loss.

As mentioned in Barachant and Congedo (2014), this invariance property has practical consequences for covariance matrices extracted from EEG signals. 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 and as noted in Harandi et al. (2014), this class of transformation includes changes of illumination when using RGB values.

Particular cases of the congruence transform when V is an SPD matrix or an orthonormal matrix have been respectively used in Yger and Sugiyama (2015) and in Harandi et al. (2014). In this paper, we also investigate invariance to orthonormal matrices. In terms of our cost function, for δPCA this means that any orthonormal subspace W will be equivalent to WO with O any matrix in the orthogonal group $\mathbb{O}_p$. Such an invariance is a particular case of a congruent transform and is naturally encoded in the definition of the Grassmann manifold $\mathcal{G}(n, p)$. This motivates our use of the Grassmann manifold for δPCA based on δ_{r} or δ_{s}.

On the other hand, the log-Euclidean metric (and the derived distance) is 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 property that for any SPD matrix A and invertible matrix V, we have $\log(VAV^{-1}) = V \log(A)V^{-1}$ (Bhatia, 2009, p.219). Then, using the fact that for any matrix $O \in \mathbb{O}_p$, $O^\top = O^{-1}$, it follows that $\delta_{\mathrm{le}}(OXO^\top, OYO^\top) = \delta_{\mathrm{le}}(X, Y)$, once again motivating the use of the Grassmann manifold.

Finally, in Eq. (4), due to the invariance of the trace to cyclic permutations[5], the product term $WW^\top$ appears twice. For any orthogonal matrix $O \in \mathbb{O}_p$, replacing W by WO in Eq. (4) will not change anything. Hence, as the Euclidean PCA (for vector data), the matrix PCA proposed in this paper is invariant under the action of the orthogonal group and explains our formulation on the Grassmann manifold.

2.5. Optimization on the Grassmann manifold

To sum up, our approach consists of finding a lower-dimensional manifold $\mathcal{S}_+^p$ by optimizing a transformation (parameterized by W) that maximizes the (approximate) Fréchet variance w.r.t. 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,

4. This property is also referred to as affine invariance.
5. That is, $\forall A, B, C, \mathrm{tr}(ABC) = \mathrm{tr}(CAB) = \mathrm{tr}(BCA)$.

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:

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, $\nabla_W f = D_W f - WW^\top D_W f$ (Absil et al., 2009).

2. Perform a line search along geodesics at W in the direction $H = \nabla_W f$. In our case, 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.

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

3. Numerical Experiments

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. Next, we apply them 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.

3.1. Synthetic data

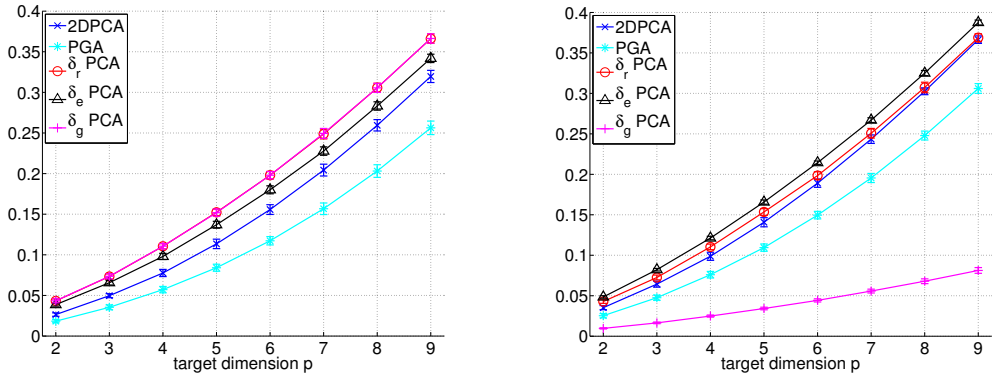
Our first goal is to substantiate the claim that our methods outperform the standard matrix PCA in terms of variance maximization. As shown in Chap. 6 of Jolliffe (2002), it is useful to study the fraction of variance retained by the method as the dimension grows. To this end we randomly generate a set $\mathbf{X} = \{X_i \in \mathcal{S}_+^n\}$ 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 $[0.5, 4.5]$. The resulting matrices are then $X_i = Q\mathrm{diag}\,(\lambda)\,Q^\top$, where each X_i has a unique matrix Q and spectrum λ.

Each matrix was compressed to size $p \times p$ for $p = 2, \ldots, 9$ using our various δPCA methods, 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. For all δPCA methods, the matrix W was initialized by the first p columns of the identity matrix. For each value of p we recorded the fraction of the Fréchet variance contained in the compressed dataset,

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

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



(a) Riemannian variance (b) Euclidean variance

Figure 3: Fraction of Fréchet variance retained by the various compression methods w.r.t. (a) the Riemannian and (b) the Euclidean distances.

We also performed this experiment with the log-Euclidean metric. However, since it is an approximation of the Riemannian metric, $\alpha_{\delta_{\text{le}}}$ exhibits essentially the same behavior as $\alpha_{\delta_{\text{r}}}$. We omit the corresponding figure for brevity.

The results of the experiment, averaged over all iterations, are presented in Fig. 3. We note that the methods δ_{s}PCA and δ_{le}PCA obtained almost identical results to δ_{r}PCA. So, for clarity, of the δPCA methods we display the results only for δ_{r}PCA, δ_{e}PCA and δ_{g}PCA. The curves of δ_{r}PCA and δ_{g}PCA coincide for the Riemannian variance.

First, with the exception of one case, our δPCA methods retain the greatest fraction of variance. As expected, each δPCA method is best at retaining the variance w.r.t. its own metric. That is, for $\alpha_{\delta_{\text{r}}}$, δ_{r}PCA outperforms δ_{e}PCA, and for $\alpha_{\delta_{\text{e}}}$ 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., $\tilde{X}_i = \bar{X}_{\text{r}}^{-1/2} X_i \bar{X}_{\text{r}}^{-1/2}$. While this transformation preserves the Riemannian distance between matrices, it does not preserve the Euclidean distance. Thus, we obtain poor results for $\alpha_{\delta_{\text{e}}}$ using this method.

3.2. Brain-computer interface

Following the promising results on the synthetic data, we next test our methods on real data. The use of covariance matrices is prevalent, for example, in the brain computer interface (BCI) community. 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 is a crucial step for building an efficient BCI system. Hence, an unsupervised dimensionality reduction method preserving 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 3 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 5 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 5^{th} order Butterworth filter. For each trial, we extracted features from the time segment located from 0.5s to 2.5s after the cue instructing the subject to perform MI.

The quality of performance of the dimensionality reduction is judged via classification error using the following scheme: 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 described and referred to as *minimum distance to the mean* (MDM) in Barachant et al. (2012) and it is restricted here to a two-classes problem with various distances.

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* (CSP). We used both the Riemannian and the Euclidean metrics to compute the class means and distances to 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 D of the supplementary material.

The accuracy rates of the classification are presented in Table 1. 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. Since we intend to extend our approach to the supervised learning setting, it is instructive to quantitatively assess the performance gap even at this early 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.

4. 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 opti-

Table 1: Accuracy rates for the various PCA methods using the Riemannian metric. The best method (excluding CSP+LDA) is highlighted by boldface.

	data set IIIa				data set IV					
Subject	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_{r}PCA	**95.56**	**68.33**	85	82.96	**55.36**	**94.64**	52.04	50.89	68.65	64.32
δ_{e}PCA	84.44	60	73.33	72.59	**55.36**	73.21	53.57	65.62	58.33	61.22
δ_{s}PCA	**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_{g}PCA	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

mization 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 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. (2014).

Acknowledgments

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

References

Pierre-Antoine Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

Awad H. Al-Mohy and Nicholas J. Higham. 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, 2009.

Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, 2006.

Vincent Arsigny, Pierre Fillard, Xavier Pennec, and Nicholas Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):328–347, 2007.

Alexandre Barachant and Marco Congedo. A Plug&Play P300 BCI using information geometry. *arXiv preprint arXiv:1409.0107*, 2014.

Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Riemannian geometry applied to BCI classification. In *Latent Variable Analysis and Signal Separation*, pages 629–636, 2010.

Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Multiclass brain–computer interface classification by Riemannian geometry. *IEEE Transactions on Biomedical Engineering*, 59(4):920–928, 2012.

Alexandre Barachant, Stéphane Bonnet, Marco Congedo, and Christian Jutten. Classification of covariance matrices using a Riemannian-based kernel for BCI applications. *Neurocomputing*, 112: 172–178, 2013.

Rajendra Bhatia. *Matrix Analysis*. Springer, 1997.

Rajendra Bhatia. *Positive Definite Matrices*. Princeton University Press, 2009.

Benjamin Blankertz, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and Klaus-Robert Müller. Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal Processing Magazine*, 25(1):41–56, 2008.

Silvere Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, Sept 2013.

Nicolas Boumal. Discrete curve fitting on manifolds. Master's thesis, Université Catholique de Louvain, jun 2010.

Nicolas Boumal and Pierre-Antoine Absil. Discrete regression methods on the cone of positive-definite matrices. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4232–4235, 2011.

Nicolas Boumal, Bamdev Mishra, Pierre-Antoine Absil, and Rodolphe Sepulchre. Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15:1455–1459, 2014. URL http://www.manopt.org.

Anoop Cherian and Suvrit Sra. Riemannian sparse coding for positive definite matrices. In *European Conference on Computer Vision*, pages 299–314, 2014.

Andrzej Cichocki, Sergio Cruces, and Shun-Ichi Amari. Log-Determinant divergences revisited: Alpha–beta and gamma log-det divergences. *arXiv preprint arXiv:1412.7146*, 2014.

Arthur P Dempster. Covariance selection. *Biometrics*, pages 157–175, 1972.

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

P. Thomas Fletcher, Conglin Lu, Stephen M. Pizer, and Sarang Joshi. Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8): 995–1005, 2004.

Maurice Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pages 215–310. Presses Universitaires de France, 1948.

Mehrtash Harandi, Mathieu Salzmann, and Richard Hartley. From manifold to manifold: geometry-aware dimensionality reduction for SPD matrices. In *European Conference on Computer Vision*, pages 17–32, 2014.

Jeffrey Ho, Yuchen Xie, and Baba Vemuri. On a nonlinear generalization of sparse coding and dictionary learning. In *International Conference on Machine Learning*, pages 1480–1488, 2013.

Stephan Huckemann, Thomas Hotz, and Axel Munk. Intrinsic shape analysis: Geodesic PCA for Riemannian manifolds modulo isometric Lie group actions. *Statistica Sinica*, 20:1–100, 2010.

Ian Jolliffe. *Principal Component Analysis*. Springer, 2002.

Fabien Lotte and Cuntai Guan. Regularizing common spatial patterns to improve BCI designs: Unified theory and new algorithms. *IEEE Transactions on Biomedical Engineering*, 58(2):355–362, 2011.

Haiping Lu, Konstantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. Multilinear principal component analysis of tensor objects for recognition. In *International Conference on Pattern Recognition*, volume 2, pages 776–779, 2006.

Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

Alois Schlögl, Felix Lee, Horst Bischof, and Gert Pfurtscheller. Characterization of four-class motor imagery EEG data for the BCI-competition 2005. *Journal of Neural Engineering*, 2(4):L14, 2005.

Stefan Sommer, François Lauze, Søren Hauberg, and Mads Nielsen. Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations. In *European Conference on Computer Vision*, pages 43–56, 2010.

Suvrit Sra. Positive definite matrices and the s-divergence. *arXiv preprint arXiv:1110.1773*, 2011.

Suvrit Sra. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *Neural Information Processing Systems*, pages 144–152, 2012.

Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: A fast descriptor for detection and classification. In *European Conference on Computer Vision*, pages 589–600, 2006.

Oncel Tuzel, Fatih Porikli, and Peter Meer. Pedestrian detection via classification on Riemannian manifolds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(10):1713–1727, 2008.

Jian Yang, David Zhang, Alejandro F. Frangi, and Jing-yu Yang. 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, 2004.

Florian Yger. A review of kernels on covariance matrices for BCI applications. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6, 2013.

Florian Yger and Masashi Sugiyama. Supervised logEuclidean metric learning for symmetric positive definite matrices. *preprint arXiv:1502.03505*, 2015.

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

Appendix A. Computing the Derivative of the Riemannian-based Cost

In Harandi et al. (2014), a cost function similar to Eq.(5) 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 $f(W) = \delta_{\mathrm{r}}^2(W^\top XW, W^\top YW)$. 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 $f(W)$:

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

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

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

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, p.60 & p.164) of the log function for $\lambda_i = \Lambda_{ii}$. The

symbol $\odot$ denotes the Hadamard product of two matrices. Then we have

$$\begin{aligned}
Df(W)[H] &= 2\left\langle D\log\circ g_{XY}(W)[H], \log\circ g_{XY}(W)\right\rangle && (14)\\
&= 2\left\langle D\log\left(g_{XY}(W)\right)\left[Dg_{XY}(W)[H]\right], \log\left(\tilde{X}^{-1}\tilde{Y}\right)\right\rangle && (15)\\
&= 2\left\langle \tilde{H}\odot\tilde{F}, V^\top \log\left(\tilde{X}^{-1}\tilde{Y}\right)V^{-\top}\right\rangle && (16)\\
&= 2\left\langle \underbrace{\left(V^\top \log\left(\tilde{X}^{-1}\tilde{Y}\right)V^{-\top}\right)\odot\tilde{F}}_{A}, \tilde{H}^\top\right\rangle && (17)\\
&= 2\left\langle VAV^{-1}, Dg_{XY}(W)[H]^\top\right\rangle && (18)\\
&= 2\left\langle \underbrace{VAV^{-1}\tilde{X}^{-1}}_{B}, W^\top YH + H^\top YW\right\rangle && (19)\\
&\quad -2\left\langle \underbrace{\tilde{X}^{-1}\tilde{Y}VAV^{-1}\tilde{X}^{-1}}_{C}, W^\top XH + H^\top XW\right\rangle\\
&= \left\langle \underbrace{2YW\left(B+B^\top\right) - 2XW\left(C+C^\top\right)}_{\nabla f(W)}, H\right\rangle, && (20)
\end{aligned}$$

where the transition between Eq.(16) and Eq.(17) 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:

$$\nabla f(W) = 2YW\left(B+B^\top\right) - 2XW\left(C+C^\top\right). \tag{21}$$

Appendix B. Computing the Derivative of the LogEuclidean-based Cost

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

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

Let us reformulate the directional derivative of f:

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

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:

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

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

$$D\left(X\mapsto X^{\top}AX\right)\left(X\right)\left[H\right]=H^{\top}AX+X^{\top}AH \tag{22}$$

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

$$\begin{aligned}
Df(W)[H] &= 2\left\langle D\log\left(W^{\top}XW\right)\left[H^{\top}XW+W^{\top}XH\right],\log\left(W^{\top}XW\right)\right\rangle && (23)\\
&+2\left\langle D\log\left(W^{\top}YW\right)\left[H^{\top}YW+W^{\top}YH\right],\log\left(W^{\top}YW\right)\right\rangle\\
&-2\left\langle D\log\left(W^{\top}XW\right)\left[H^{\top}XW+W^{\top}XH\right],\log\left(W^{\top}YW\right)\right\rangle\\
&-2\left\langle D\log\left(W^{\top}YW\right)\left[H^{\top}YW+W^{\top}YH\right],\log\left(W^{\top}XW\right)\right\rangle\\
&=2\left\langle D\log\left(W^{\top}XW\right)\left[H^{\top}XW+W^{\top}XH\right],\log\left(W^{\top}XW\right)-\log\left(W^{\top}YW\right)\right\rangle && (24)\\
&+2\left\langle D\log\left(W^{\top}YW\right)\left[H^{\top}YW+W^{\top}YH\right],\log\left(W^{\top}YW\right)-\log\left(W^{\top}XW\right)\right\rangle\\
&=2\left\langle D\log\left(W^{\top}XW\right)\left[\log\left(W^{\top}XW\right)-\log\left(W^{\top}YW\right)\right],H^{\top}XW+W^{\top}XH\right\rangle && (25)\\
&+2\left\langle D\log\left(W^{\top}YW\right)\left[\log\left(W^{\top}YW\right)-\log\left(W^{\top}XW\right)\right],H^{\top}YW+W^{\top}YH\right\rangle\\
Df(W)[H] &= \left\langle 4XWD\log\left(W^{\top}XW\right)\left[\log\left(W^{\top}XW\right)-\log\left(W^{\top}YW\right)\right],H\right\rangle && (26)\\
&+\left\langle 4YWD\log\left(W^{\top}YW\right)\left[\log\left(W^{\top}YW\right)-\log\left(W^{\top}XW\right)\right],H\right\rangle
\end{aligned}$$

From the expression of the function f, we first apply the product rule and the chain rule in order to obtain Eq. 23. Then, from Eq. 24 to Eq. 25, we use the property that $Dlog(X)[.]$ is an auto-adjoint operator[6] 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 :

$$\begin{aligned}
\nabla f(W) &= 4XWD\log\left(W^{\top}XW\right)\left[\log\left(W^{\top}XW\right)-\log\left(W^{\top}YW\right)\right] && (27)\\
&+4YWD\log\left(W^{\top}YW\right)\left[\log\left(W^{\top}YW\right)-\log\left(W^{\top}XW\right)\right]
\end{aligned}$$

Appendix C. Computing the Derivative of the Euclidean-based Cost

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

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

As for the logEuclidean case, we reformulate the cost :

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

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

Table 2: Accuracy rates for the various PCA methods using the Euclidean metric

	data set IIIa				data set IV					
Subject	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_{r}PCA	**86.67**	**61.67**	73.33	**73.89**	50	**83.93**	52.04	56.25	**76.59**	**63.76**
δ_{e}PCA	61.11	48.33	55	54.81	47.32	64.29	**54.59**	**62.5**	41.67	54.07
δ_{s}PCA	**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_{g}PCA	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

Then, reusing Eq. (22) for the directionnal derivative for the quadratic term $W^\top XW$ and making use of the composition and product rules (defined in the previous section), we have :

$$\begin{aligned} Df(W)[H] &= 2\left\langle H^\top XW + W^\top XH, W^\top XW\right\rangle + 2\left\langle H^\top YW + W^\top YH, W^\top YW\right\rangle \\ &\quad - 2\left\langle H^\top XW + W^\top XH, W^\top YW\right\rangle - 2\left\langle H^\top YW + W^\top YH, W^\top XW\right\rangle \\ &= 2\left\langle H^\top XW + W^\top XH, W^\top (X-Y) W\right\rangle + 2\left\langle H^\top YW + W^\top YH, W^\top (Y-X) W\right\rangle \\ &= 4\left\langle H^\top XW, W^\top (X-Y) W\right\rangle - 4\left\langle H^\top YW, W^\top (X-Y) W\right\rangle \\ &= 4\left\langle H^\top (X-Y) W, W^\top (X-Y) W\right\rangle \end{aligned} \tag{28}$$

$$Df(W)[H] = \left\langle 4 (X-Y) WW^\top (X-Y) W, H\right\rangle \tag{29}$$

Hence, we have :

$$\nabla f(W) = 4 (X-Y) WW^\top (X-Y) W \tag{30}$$

Appendix D. BCI classification results using Euclidean metric

Table 2 contains the results of BCI data classification using the MDM classifier with the Euclidean metric.