Riemannian Procrustes Analysis: Transfer Learning for Brain-Computer Interfaces
Pedro Luiz Coelho Rodrigues, Christian Jutten, Marco Congedo

To cite this version:

HAL Id: hal-01971856
https://hal.archives-ouvertes.fr/hal-01971856
Submitted on 7 Jan 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Riemannian Procrustes Analysis: Transfer Learning for Brain-Computer Interfaces

Pedro L. C. Rodrigues, Member, IEEE, Christian Jutten, Fellow, IEEE, and Marco Congedo, Member, IEEE

Abstract—Objective: This paper presents a Transfer Learning approach for dealing with the statistical variability of EEG signals recorded on different sessions and/or from different subjects. This is a common problem faced by Brain-Computer Interfaces (BCI) and poses a challenge for systems that try to reuse data from previous recordings to avoid a calibration phase for new users or new sessions for the same user. Method: We propose a method based on Procrustes analysis for matching the statistical distributions of two datasets using simple geometrical transformations (translation, scaling and rotation) over the data points. We use symmetric positive definite matrices (SPD) as statistical features for describing the EEG signals, so the geometrical operations on the data points respect the intrinsic geometry of the SPD manifold. Because of its geometry-aware nature, we call our method the Riemannian Procrustes Analysis (RPA). We assess the improvement in Transfer Learning via RPA by performing classification tasks on simulated data and on eight publicly available BCI datasets covering three experimental paradigms (243 subjects in total). Results: Our results show that the classification accuracy with RPA is superior in comparison to other geometry-aware methods proposed in the literature. We also observe improvements in ensemble classification strategies when the statistics of the datasets are matched via RPA. Conclusion and significance: We present a simple yet powerful method for matching the statistical distributions of two datasets, thus paving the way to BCI systems capable of reusing data from previous sessions and avoid the need of a calibration procedure.

Index Terms—Brain-Computer Interface, Riemannian geometry, Transfer Learning, Covariance Matrices, EEG.

I. INTRODUCTION

A Brain-Computer Interface (BCI) is a system that allows a person to interact with a machine without any physical interaction. It works by extracting features from neuro-physiological signals (e.g., the power spectral densities on certain frequency bands) and assigning them to different classes. These classes may be associated to cognitive states, sensory responses, etc., and the features are chosen so that they are discriminative for each class. Among the many types of neural signals that have been used in BCI systems, electroencephalographic (EEG) recordings have received particular attention and are the focus of this paper. The success of EEG-based systems comes from several reasons, such as its low price as compared to other neuro-physiological modalities, its non-invasiveness, and its high temporal resolution. However, they also have to cope with low signal-to-noise ratio, low spatial resolution [1], and high cross-session and cross-subject variability [2].

While many signal processing techniques have been developed to ameliorate the quality of EEG signals and improve its spatial resolution [3], the cross-session and cross-subject variability problem have received much less attention. The standard approach in BCI consists of re-training a statistical classifier at the beginning of every experimental session with the help of a sequence of calibration trials, a procedure that can be time consuming and is clearly suboptimal, since it does not leverage any information from past experiments. Instead, “second-generation BCIs” initialize classifiers using Transfer Learning and update their parameters along sessions, avoiding the calibration step altogether [1].

We present in this paper a Transfer Learning approach based on geometrical transformations for matching the statistical distributions coming from two different experimental sessions, a source and a target session. For simplicity of exposition, hereafter we use the term “session” generically, however source and target may refer to different sessions as well as different subjects. We perform simple linear transformations, such as translation, rotation, and scaling, on the data points of both sessions with the goal of making the shape of their statistical distributions as similar as possible. Once the distributions are matched, one can expect that a classifier optimized for the data of the source session will work well enough with the data from the target session.

The branch of Machine Learning that studies the effects of mismatches between statistical distributions is called Transfer Learning [4]. It has been of great interest in several domains besides BCI, such as in computer vision, where the statistics of the data may vary due to changes in lighting conditions and acquisition devices, or in speech processing systems, where the changes in background noise and the differences in speaker genders and voice tonalities may affect the statistics of the signals. In [5], the phenomenon responsible for the drift in statistical distributions of two datasets was termed covariate shift and modelled by assuming that the distributions of the data points can be different for the source and target datasets, but the conditional distributions of the labels are the same. Ref. [6] presented real data examples on BCI experiments and showed that the covariate shift describes well the changes in statistics for this kind of application. A recent attempt in the literature on Transfer Learning has been to apply the theory of Optimal Transport to determine which geometrical transformations one should perform to match two statistical distributions, as proposed in [7] and applied to the BCI P300 paradigm in [8]. This approach begins by first defining a
function that models the cost of transporting a point at location \( x \) to a location \( y \). Then, it solves an optimization problem that minimizes the total cost of moving each point of the source dataset and make its statistical distribution as similar as possible to the distribution of the target dataset. Note that this is a completely unsupervised procedure and does not rely on any assumption regarding the statistical distributions of the datasets.

Another geometrical approach [9] matches the statistical distributions from the source and target sessions by means of a re-centering of their data points to the origin of the SPD space, the identity matrix. During the process of review of this paper, we came across the work in [10], which also proposes a Transfer Learning procedure for SPD matrices in the spirit of [9], but with the main difference that the data points are re-centered to the midpoint between the geometric means of the source and target datasets.

Besides of distribution matching based on geometrical transformations, there has been mainly two other kinds of proposals for doing Transfer Learning in the BCI literature [2]. One is based on the concept of ensemble classifiers [3], [11]–[13], where the information from multiple source datasets are combined into a “global” classifier, which is then used to label the trials from any other target dataset. There has also been works using Bayesian models to describe the variability of the statistics on the source datasets and gather information from multiple datasets [14]. A recent approach that builds upon such Bayesian methods are the works in [15] and [16], which use a special form of the P300 experimental paradigm to do classification with no calibration. There are two main differences between these approaches and our proposal in this work. First, our approach is based on matching as much as possible the information from each source-target pair of datasets. Thus, it can be used in addition to an ensembling approach as in [12], which will combine the information from multiple matched-source subjects. Second, our approach is paradigm-agnostic and does not rely on any special modification of the experimental setup where the EEG signals are collected (as opposed to [16]), a feature that is appealing to a great number of practitioners.

Our approach for distribution matching is based on the concept of Procrustes Analysis (PA) [17], a tool often employed in statistical shape analysis [18] with applications in text analysis [19], protein alignment [19], and many other fields. PA works by first selecting a set of pairs of landmark points from two different shapes and then performing geometrical transformations to get these landmarks as close as possible to each other. In our context, the shapes to be matched are actually point clouds consisting of high-dimensional statistical features describing EEG signals, and the landmarks are points that can be used to describe these statistical distributions (e.g. the mean, the farthest point from the mean, etc.). Because of its linear nature in the Euclidean case, and the fact that the operations are always global (the same rotation/translation/scaling is applied to all points each time), the space of transformations that one can cover using Procrustes Analysis does not include all possible transformations between the statistics of datasets. Nevertheless, the results that we have obtained on BCI applications indicate that the set of transformations that we apply are rich enough to model the covariate shift between experimental sessions.

A particularity of our work is that we use the spatial covariance matrices (SCM) of the EEG signals as statistical features to discriminate between classes. SCMs are symmetric positive definite matrices (SPD) and as such they are defined in a Riemannian manifold [20] whose geometry is taken into consideration during the classification procedure. Such an approach has proved successful in BCI applications in recent years [21]–[23] and has become part of the state of the art of the field [2]. The geometric transformations that we use for matching two statistical distributions are done taking into account the intrinsic geometry of the SPD manifold as well. Because of its geometry-aware nature, we call our method Riemannian Procrustes Analysis (RPA).

RPA can be seen as an evolution of the aforementioned procedures [9] and [10], with the re-centering step corresponding to the first of a series of geometrical transformations. Furthermore, the procedures in [9] and [10] are completely unsupervised, since they do not use any information from the labels of the data points, whereas RPA benefits from the labels in the source session (which are all known in advance) as well as from (at least part of) the labels that become sequentially available in the target session trial after trial.

We compared the performance of the RPA procedure for Transfer Learning to that of other distribution-matching methods proposed in [9], [10] and [7]. The results demonstrate that the RPA yields a superior classification accuracy in both simulated and real datasets. We also observe that, in general, RPA needs a very small amount of labeled trials from the target dataset to work well. Moreover, it is always superior (or at least equal) to the performance obtained with calibration, i.e., simply using the available trials on the target dataset and not do any transfer from the source.

The remainder of the paper goes as follows: Section II gives a mathematical formulation for the paradigm of Transfer Learning that we consider in this work. Section III introduces the tools for manipulating and classifying SPD matrices in their Riemannian manifold, and Section IV presents the method of Riemannian Procrustes Analysis. Sections V and VI discuss the results on simulated and real data, respectively. Lastly, Section VII concludes the paper.

II. PROBLEM FORMULATION

We consider two datasets, the source (\( S \)) and the target (\( T \)) datasets. They are comprised of couples

\[
S = \left\{ (C_i, y_i) \text{ for } i = 1, \ldots, K_S \right\},
\]

\[
T = \left\{ (\tilde{C}_i, \tilde{y}_i) \text{ for } i = 1, \ldots, K_T \right\},
\]

with \( C_i, \tilde{C}_i \in \mathbb{R}^{n \times n} \) being data points, and \( y_i, \tilde{y}_i \in \{1, \ldots, L\} \) their corresponding class labels; \( K_S \) and \( K_T \) are the number of trials in the source and target sessions respectively. In this paper, the data points in \( S \) and \( T \) are always symmetric positive definite (SPD) matrices and are used to parametrize EEG multivariate time series [22]. As mentioned before,
SPD matrices are defined in a Riemannian manifold, so it is important that operations involving them respect the intrinsic geometry of this space (see Section III for more information).

Our distribution matching method considers the semi-supervised Transfer Learning paradigm [4], where one has knowledge of all the labels from the source dataset and access to a small subset of labels from the target dataset. Put in mathematical terms, we assume knowledge of all the labels from the elements in $S$ and of a small subset $T_\ell \subset T$ with

$$T = T_\ell \cup T_u \quad \text{and} \quad T_\ell \cap T_u = \emptyset,$$

where $\ell$ stands for labeled and $u$ for unlabeled. We further assume that $T_\ell$ has at least one example from each class. This setup describes well applications where a few calibration points from the target dataset can be used to guide the Transfer Learning procedure. Another relevant case is online algorithms, where labels are available sequentially and augment the $T_\ell$ dataset after each time step.

Our goal is to train a classifier that leverages the available information from $S$ and $T_\ell$ and has good accuracy on the classification of data points from $T_u$.

### III. BACKGROUND AND NOTATION

This section begins with a brief introduction to concepts of Riemannian geometry on SPD matrices. Then, we define the notion of statistical distributions of SPD matrices and review Riemannian geometry on SPD matrices. Then, we define the geometric mean of a set of points, etc.

#### A. The Symmetric Positive Definite manifold

Let $\mathcal{P}(n)$ be the set of $n \times n$ symmetric positive definite (SPD) matrices, which is defined as

$$\mathcal{P}(n) = \left\{ C \in \mathbb{R}^{n \times n} \mid C^T = C, \ x^T C x > 0, \ \forall x \in \mathbb{R}^n \right\}.$$

Matrices in $\mathcal{P}(n)$ lie in a manifold [20], a set of points with the property that the neighborhood of each $C \in \mathcal{P}(n)$ can be bijectively mapped to an Euclidean space, also known as its tangent space $T_C : \mathcal{P}(n)$. In particular, because $\mathcal{P}(n)$ is an open subspace of the set $\text{Sym}(n)$ of symmetric matrices in $\mathbb{R}^{n \times n}$, we can identify its tangent space as simply being $\text{Sym}(n)$ [24]. If we endow every tangent space of a manifold with a metric that changes smoothly along its elements, we say that we have a Riemannian manifold [24]. In this case, fundamental geometric notions are naturally defined, such as geodesic (shortest curve joining two points), distance between two points (length of the geodesic connecting them), the center of mass of a set of points, etc.

There are several possible choices of metric for $\mathcal{P}(n)$ and each one induces a different geometry that can be more or less adequate according to the applications we are interested in. A metric that is particularly relevant is the Affine-Invariant Riemannian metric (AIMR) [20], defined for $\eta, \xi \in T_C : \mathcal{P}(n)$ as the inner product

$$\langle \eta, \xi \rangle_C = \text{tr}(C^{-1}\eta C^{-1}\xi) ,$$

where $C \in \mathcal{P}(n)$ and $\text{tr}(\cdot)$ denotes the trace operator. It is well known [20] that the distance between matrices $C_i, C_j \in \mathcal{P}(n)$ induced by (4) is

$$\delta_R^2(C_i, C_j) = \sum_{k=1}^{n} \log^2(\lambda_k) ,$$

where the $\lambda_k$’s are the eigenvalues of $C_i^{-1}C_j$ or of similar matrix $C_i^{-\frac{1}{2}}C_j C_i^{-\frac{1}{2}}$. Because of the many interesting properties of (5), such as invariance to affine transformations by any invertible matrix $A \in \mathbb{R}^{n \times n}$,

$$\delta_R^2(C_i, C_j) = \delta_R^2(AC_iA^T, AC_jA^T) ,$$

invariance under inversion, etc. [20], the AIRM has found great popularity in geometry-aware algorithms for processing SPD matrices [21] [25]. From now on, whenever we refer to $\mathcal{P}(n)$ we will be implicitly considering it has been equipped with the AIRM.

The center of mass according to distance (5) of a set of SPD matrices $\{C_1, \ldots, C_K\}$ is defined as [1]

$$\mathcal{G} (\{C_i\}_{i=1}^K) = \arg\min_{X \in \mathcal{P}(n)} \sum_{i=1}^{K} \delta_R^2(X, C_i) .$$

In words, $\mathcal{G} (\{C_i\}_{i=1}^K)$ is the point in the manifold minimizing the dispersion (variance) of the set of matrices. Note that when the $C_k$’s are actually strictly positive scalars, $\mathcal{G} (\{C_i\}_{i=1}^K)$ is their geometric mean. This explains why many researchers adopt the term “geometric mean” to refer to the center of mass of a set of SPD matrices. For three or more matrices, there is no closed form solution for (7), so one has to resort to iterative algorithms as the one given in [26]. The above definitions suffice for the intents of this paper, but the interested reader will find a thorough treatment of the subject in [20].

#### B. Statistical distributions in the SPD manifold

In this work, we assume that the data points come from statistical distributions that can be parametrized just by their geometric mean and the dispersion of points around it. More precisely, we assume that the statistical distribution generating the samples of the dataset is a mixture of Riemannian Gaussian distributions on the SPD manifold (one for each class) [27].

Under this assumption, we parametrize the statistics of each dataset defined in (1) using a set consisting of $L+2$ elements : the geometric mean $M$ of the dataset, the geometric means $M_k$ of each of the $L$ classes, and the dispersion $d$ around $M$. We have then

$$\Theta_S = \{ M, M_1, \ldots, M_L, \ d \} ,$$

$$\Theta_T = \{ \tilde{M}, \tilde{M}_1, \ldots, \tilde{M}_L, \ d \} ,$$

where

$$M = \mathcal{G} (\{ C_i \mid C_i \in \mathcal{S} \}) ,$$

$$\tilde{M} = \mathcal{G} (\{ \tilde{C_i} \mid \tilde{C_i} \in T_\ell \})$$
are the center of mass or geometric means of the datasets (all classes combined),
\[
d = \sum_{C_i \in S} \delta^2_R(M, C_i),
\]
\[
\tilde{d} = \sum_{C_i \in T_i} \delta^2_R(\tilde{M}, C_i)
\]
are the dispersions around the geometric mean, and
\[
\tilde{M}_s = \mathcal{G}(\{C_i \mid C_i \in S \text{ and } y_i = k\}),
\]
\[
\tilde{M}_k = \mathcal{G}(\{\tilde{C}_i \mid \tilde{C}_i \in T_i \text{ and } \tilde{y}_i = k\})
\]
are the geometric mean of the trials belonging to each class. Note that the parameters for the target dataset are estimated using only the data points from \(T_i\).

The law of large numbers from statistics also applies to datasets containing SPD matrices [20]. In particular, if the elements of a dataset come from a statistical distribution with geometric mean \(M\), the center of mass of a set of \(K\) matrices will converge to \(M\) as \(K\) grows. This implies that in experimental paradigms where the trials come in sequentially, it is reasonable to expect that with more trials one will obtain better and better estimates of the geometric mean.

C. Classification in the SPD manifold

There are many known classifiers for datasets whose elements are SPD matrices [22]. In this paper, we use the Minimum-Distance to Mean (MDM) classifier [21], which works as follows: in the training phase, calculate the geometric means for each class of a training dataset \(D_{\text{train}}\). In the testing phase, each matrix \(C_i\) from the testing dataset \(D_{\text{test}}\) is associated to the label \(y_i\) of the class mean that is the closest to \(C_i\). Note that the statistical model described in Section III-B fits well the MDM classifier, since it assumes that the statistics of the dataset can be described by the geometric means of its classes.

IV. TRANSFER LEARNING VIA PROCRUSTES ANALYSIS

This section presents the Riemannian version of the Procrustes Analysis or RPA. We first introduce the concept of Procrustes Analysis on an Euclidean setting and then describe how to perform equivalent transformations on the SPD manifold. Then, we justify such operations with the help of a model relating the statistical distributions of the source and target datasets. Such model is the main theoretical contribution of this paper, since it gives a concrete justification for the geometric operations done in the RPA procedure and allows for a better comprehension of the assumptions that one has to make regarding the statistical distributions of the datasets.

A. Matching statistical distributions

A common approach to match two shapes in Euclidean space is the Procrustes Analysis [17], which works as follows: suppose we have two sets of landmark points
\[
\mathcal{X} = \{x_i \in \mathbb{R}^n\}_{i=1}^m \text{ and } \tilde{\mathcal{X}} = \{\tilde{x}_i \in \mathbb{R}^n\}_{i=1}^m,
\]
and assume there is a linear relationship relating the \(m\) pairs of landmark points as in
\[
\tilde{x}_i - m = s U (x_i - m),
\]
where \(s \in \mathbb{R}, m, \tilde{m} \in \mathbb{R}^n, \text{ and } U \in \mathbb{R}^{n \times n}\) is an orthogonal matrix. The goal of the procedure is to determine the values of \(\{s, m, \tilde{m}, U\}\) so to obtain a new set \(\tilde{\mathcal{X}}^{(PA)}\) containing points \(\tilde{x}_i^{(PA)}\) that matches exactly with \(x_i\), where
\[
\tilde{x}_i^{(PA)} - \tilde{m} = \frac{1}{s} U^T (\tilde{x}_i - \tilde{m}).
\]
Note that the operations transforming \(\tilde{x}_i\) can be interpreted as a re-centering to zero (subtracting \(\tilde{m}\)) followed by a stretching or compression (division by \(s\)), and a rotation (multiplication by \(U^T\)); the final re-centering to \(m\) is optional, since it is often more interesting to re-center \(\mathcal{X}\) to the origin and consider only the zero-mean matched shapes.

B. Riemannian Procrustes analysis (RPA)

In order to do Procrustes Analysis on SPD matrices, we have to adapt the steps of re-centering, stretching and rotation according to the intrinsic geometry of \(\mathcal{P}(n)\). We call such procedure Riemannian Procrustes Analysis (RPA) and describe its steps here below:

1) Re-center matrices to identity: In \(\mathcal{P}(n)\) the Identity matrix plays the role of the origin of the space. Therefore, the first step of RPA is to transform the matrices in \(S\) and \(T\) so they are both centered around \(I_n\) (see Fig. 1B). This amounts to the transformation proposed in [9] if the covariance matrices used to describe the resting activity of each session were chosen to be the geometric mean of the trials of each dataset.

Due to the affine-invariance of (5) and (7), the geometric mean of the set of re-centered matrices
\[
C_i^{(\text{rect})} = M^{-1/2} C_i M^{-1/2}
\]
is \(I_n\). Moreover, since \(\tilde{M}\) is estimated from a subset of points in \(T_i \subset T\), the geometric mean of the set of matrices
\[
\tilde{C}_i^{(\text{rect})} = \tilde{M}^{-1/2} \tilde{C}_i \tilde{M}^{-1/2}
\]
is approximately the identity matrix (it tends to the identity as the number of elements in \(T_i\) grows).

We have then two new datasets consisting of re-centered matrices
\[
S^{(\text{rect})} = \left\{ (C_i^{(\text{rect})}, y_i) \text{ for } i = 1, \ldots, K_S \right\},
\]
\[
T^{(\text{rect})} = \left\{ (\tilde{C}_i^{(\text{rect})}, \tilde{y}_i) \text{ for } i = 1, \ldots, K_T \right\},
\]
with the indices of the partition \(T^{(\text{rect})} = T_i^{(\text{rect})} \cup T_n^{(\text{rect})}\) being the same as in (1).

2) Equalize the dispersions on each dataset: The next step of RPA consists of rescaling the distributions on both datasets so that their dispersions around the mean are the same (see Fig. 1C). To do so, we can see from (5) that
\[
\delta^2_R (\tilde{C}_i^{(\text{rect})}, I_n) = s^2 \delta^2_R (C_i^{(\text{rect})}, I_n),
\]
which implies that one can modulate the dispersion of $T^{\text{rot}}$ by simply moving each of its matrices along the geodesic that links it to the identity matrix. Note that the parameter $s$ plays the same role of the scaling factor in (9). We match the dispersions from source and target by building a new dataset $T^{\text{str}}$ containing the stretched matrices

$$\tilde{C}_i^{\text{str}} = (\tilde{C}_i^{\text{rot}})^s,$$

where we require $s \in \mathbb{R}$ to verify

$$s^2 = d/\tilde{d}.$$  

Note that the re-centering of matrices in Step 1 does not alter the dispersion of the matrices around their geometric mean, which means that the stretching step could have been done before re-centering the matrices in $T$. However, in this case, the geodesic move in (15) would have to be done with respect to $\tilde{M}$, that is, we would have to use a more involved relation

$$\tilde{C}_i^{\text{str}} = \tilde{M}^{1/2} \left( \tilde{M}^{-1/2} \tilde{C}_i \tilde{M}^{-1/2} \right)^s \tilde{M}^{1/2}.$$  

Note that up to this point no information from the trials’ classes has been used. We say then that the recentering and stretching operations form the unsupervised part of the RPA method.

3) Rotate around the geometric mean: The last step consists of rotating the matrices from $T^{\text{str}}$ around the origin and matching the orientation of its point cloud with that of $S^{\text{rot}}$ (see Fig. 1D). To do so, we note that if $U$ is an orthogonal matrix, then

$$\delta_R^2(U^T \tilde{C}_i^{\text{str}} U, U^T U) = \delta_R^2(\tilde{C}_i^{\text{str}}, I_n),$$

which indicates that the effect of an orthogonal matrix over a set of matrices centered at the identity is that of a rotation around their mean. We form a new dataset $T^{\text{rot}}$ containing rotated matrices with

$$\tilde{C}_i^{\text{rot}} = U^T \tilde{C}_i^{\text{str}} U,$$

where $U$ is an orthogonal matrix to be determined from the data. As we will see in the next sub-section, matrix $U$ is determined using the labels from the trials, so it corresponds to the supervised part of the procedure.

C. The orthogonal matrix $U$

The procedure to determine matrix $U$ comes up naturally once the assumptions of the RPA method are written in mathematical form. For simplicity, we will first assume that $T_0 = T$. We use the geometric means of the source and target datasets as landmarks to be matched, so one can write

$$\tilde{M} = A M A^T,$$

and

$$\tilde{M}_k = A M_k A^T \quad k \in \{1, \ldots, L\},$$

where $M, \tilde{M}, M_k, \tilde{M}_k$ are all defined in Section III-B, and $A \in \mathbb{R}^{n \times n}$ is an unknown invertible matrix that models the discrepancies between the statistics of the source and target datasets. We can rewrite the relation in (20) as

$$\left( \tilde{M}^{1/2} \tilde{M}^{1/2} \right) = A \left( M^{1/2} M^{1/2} \right) A^T,$$

and

$$I_n = \tilde{M}^{-1/2} A \left( M^{1/2} M^{1/2} \right) A^T \tilde{M}^{-1/2},$$

$$I_n = \left( \tilde{M}^{-1/2} A M^{1/2} \right) \left( \tilde{M}^{-1/2} A M^{1/2} \right)^T,$$

$$U^T U = \left( \tilde{M}^{-1/2} A M^{1/2} \right) \left( \tilde{M}^{-1/2} A M^{1/2} \right)^T,$$

where $U$ is the $n \times n$ orthogonal matrix that we want to determine. Matrix $U$ can then be simply written as

$$U = \tilde{M}^{-1/2} A M^{1/2},$$

where $M$ and $\tilde{M}$ are directly estimated from the data points as explained in Section III-A, and $A$ remains unknown. To determine an expression for $U$ only in terms of variables that can be estimated from the data, we use (26) in (21) to get

$$\tilde{M}_k = \left( \tilde{M}^{1/2} U M^{-1/2} \right) M_k \left( \tilde{M}^{1/2} U M^{-1/2} \right)^T,$$

and using the expression in (20) for $\tilde{M}$, one can rewrite (28) as

$$\tilde{M}^{-1/2} \tilde{M}_k \tilde{M}^{-1/2} = U M^{-1/2} M_k M^{-1/2} U^T.$$  

Defining the matrices

$$G_k = M^{-1/2} M_k M^{-1/2}$$

and

$$\tilde{G}_k = \tilde{S}^{-1/2} \tilde{M}_k \tilde{S}^{-1/2},$$

and using the expression in (20) for $\tilde{M}$, one can rewrite (28) as

$$\tilde{G}_k = \tilde{M}^{1/2} A^{-T} M^{-1/2} G_k M^{1/2} A^T \tilde{M}^{-1/2},$$

and conclude that $G_k$ and $\tilde{G}_k$ are related via a similarity transform. Similar matrices having the same eigenvalues, one can write the eigendecompositions

$$G_k = Q_k \Lambda \tilde{Q}_k^T$$

and

$$\tilde{G}_k = \tilde{Q}_k \Lambda Q_k^T$$

so that (28) becomes

$$\tilde{Q}_k \Lambda \tilde{Q}_k^T = U Q_k \Lambda Q_k^T U^T.$$  

Solving (32) for $U$ we obtain

$$U = \tilde{Q}_k Q_k^T,$$

which is ultimately an expression in terms of variables estimated from the dataset. Note that (33) is also the solution to the following optimization problem

$$\min_{U^T U = I_n} \delta_R^2(\tilde{G}_k, U G_k U^T),$$

and so it can be interpreted as the orthogonal matrix that acts to minimize the distance between a modified version of the class means of the source and target datasets.
D. Determining \( U \) from data

In the previous section, we assumed \( T_\ell = T \). In practice, however, we have \( T_\ell \subset T \), so the estimation of the class means of the target dataset are only approximations of the real class means of the statistical distribution. Because of this, instead of giving preference to a particular noisy estimate of a class mean to determine \( U \) via (33), we obtain it as a solution to the following optimization problem on the manifold of orthogonal matrices:

\[
\minimize_{U^T U = I_u} \sum_{k=1}^{L} w_k \delta_R^2 \left( \tilde{G}_k, UG_k U^T \right), \quad (35)
\]

where the \( w_k \in [0, 1] \) are coefficients allowing to balance the optimization according to the quality of the estimators for the mean of each class. Note that problem (35) is a generalization of the Procrustes problem in the SPD manifold proposed in [28]. We solve (35) using a special form of the steepest-descent algorithm adapted for optimization procedures on manifolds, as described in [24]. To do so, we first rewrite each term of the cost function in (35) as

\[
f_k(U) = \delta_R^2 \left( \tilde{G}_k, UG_k U^T \right) \quad (36)
\]

and express its Jacobian as

\[
D_U L(U) = \sum_{k=1}^{L} w_k \ D_U f_k(U), \quad (37)
\]

with

\[
D_U f_k(U) = 4 \log \left( \tilde{G}_k UG_k U^T \right) U, \quad (38)
\]

where the derivative of the AIRM distance was obtained from [29]. On each iteration of the gradient descent procedure, the vector \( D_U f_k(U) \) is projected onto the tangent space of the manifold of orthogonal matrices (see [24] for details). We used the pymanopt package [30] for carrying out the optimization procedure.

E. Classification on the transformed datasets

Once the datasets have been transformed with RPA, we use the MDM classifier (see Section III-C) to infer the unknown labels from the elements in \( T_u \). In the training phase we have:

\[
D_{\text{train}} = S^{(\text{rot})} \cup \left\{ \left( \tilde{C}_i^{(\text{rot})}, \tilde{y}_i \right) \mid (\tilde{C}_i, \tilde{y}_i) \in T_\ell \right\}, \quad (39)
\]

and in the testing phase we infer the labels from

\[
D_{\text{test}} = \left\{ \left( \tilde{C}_i^{(\text{rot})}, \tilde{y}_i \right) \mid (\tilde{C}_i, \tilde{y}_i) \in T_u \right\}. \quad (40)
\]

F. Summary of the RPA method

Algorithm 1 recapitulates the steps of a classification task using RPA for matching the statistical distributions of the source and target datasets.

**Algorithm 1: Transfer Learning via RPA**

**Input:** \( S, T_\ell \) and \( T_u \) as defined in (1) and (2)

**Output:** accuracy of classification using MDM on \( T_u \)

1. Estimate \( M \) and \( \hat{M} \) from the data in \( S \) and \( T_\ell \)
2. Re-center the matrices in \( S \) and \( T \) using (11) and (12), and form new datasets \( S^{(\text{rot})} \) and \( T^{(\text{rot})} = T^{(\text{rot})} \cup T_u^{(\text{rot})} \)
3. Calculate the ratio of dispersions in \( S^{(\text{rot})} \) and \( T^{(\text{rot})} \) as in (16) and use it to form the new dataset \( T^{(\text{rot})} = T_u^{(\text{rot})} \cup T_u^{(\text{rot})} \)
4. Estimate matrices \( M_k \) and \( \hat{M}_k \) for \( k \in \{1, \ldots, L\} \) and obtain the orthogonal matrix \( U \) as a solution from (35)
5. Rotate the matrices from \( T_u^{(\text{rot})} \) as in (19) and obtain \( T_u^{(\text{rot})} \)
6. Form the training dataset for the MDM classifier with \( D_{\text{train}} = S^{(\text{rot})} \cup T_u^{(\text{rot})} \)

and get the accuracy of classification on the data points from the test dataset

\[
D_{\text{test}} = T_u^{(\text{rot})}
\]

G. An interpretation of the steps in RPA

We give now an interpretation of the steps of RPA in terms of the statistical distributions of the datasets. Without loss of generality, we will consider that \( d = d \), since the dispersions can always be made equal prior to the transformations. We will also assume \( T_\ell = T \) for simplicity.

The relations in (20) and (21) define which landmark data points we should match in the RPA procedure, an approach that is justified from the fact that we parametrize the statistics of \( S \) and \( T \) using their geometric means, as described in Section III-B. From this observation, one can also conclude that a simple approach for matching the statistical distributions of \( S \) and \( T \) would be to estimate matrix \( A \) from the available data points and apply \( A^{-1} \) to all the elements of \( T \), as in

\[
\tilde{C}_i \mapsto A^{-1} \tilde{C}_i A^{-T}. \quad (41)
\]

From (26) we can write

\[
A = \hat{M}^{1/2} U M^{-1/2}, \quad (42)
\]

where \( M \) and \( \hat{M} \) are estimated directly from the dataset, and the orthogonal matrix \( U \) is determined as discussed in Section IV-C. Applying \( A^{-1} \) to the matrices in \( T \), we get

\[
A^{-1} \tilde{C}_i A^{-T} = M^{1/2} U^T \left( \hat{M}^{-1/2} \tilde{C}_i \hat{M}^{-1/2} \right) U M^{1/2}, \quad (43)
\]

which describes the same steps of RPA: re-center to identity, stretch (with \( s = 1 \)) and rotate, followed by a translation of the mean back to \( M \). From the expressions above, we see that the sequence of operations in RPA are nicely justified by the assumptions of our statistical model for the data points.
V. NUMERICAL ILLUSTRATIONS: SIMULATED DATA

A. The dataset

Our example on simulated data consists of a source and a target datasets containing \(2 \times 2\) SPD matrices and belonging to two classes. Data points from the source dataset are generated as follows:

1) Generate a random SPD matrix \(M_1 \in \mathcal{P}(2)\) and define it to be the geometric mean of class 1
2) Generate \(N_t = 100\) random SPD matrices around \(M_1\) by mapping small random tangent vectors (norm fixed to \(\varepsilon = 10\)) from \(T_{M_1}\mathcal{P}(2)\) back to the SPD manifold. We associate to each of these matrices the label \(y_i = 1\)
3) Generate a random SPD matrix \(M_2\) whose distance to \(M_1\) is \(s = 5\). This is the geometric mean for class 2
4) Generate \(N_t = 100\) random SPD matrices around \(M_2\) by mapping small random tangent vectors (norm fixed to \(\varepsilon = 10\)) from \(T_{M_2}\mathcal{P}(2)\) back to the SPD manifold. These matrices have a label \(y_i = 2\) associated to them.

We generate the data points from the target dataset \(\mathcal{T}\) exactly as for the source dataset, but add an extra translation step that ensures that the geometric mean of all the matrices from \(S\) will be at a distance \(d = 8\) from the geometric mean of \(\mathcal{T}\).

B. Visualization of the steps of RPA

We use the algorithm of Diffusion maps [31] to obtain new representations of our data points using only two axis. This nonlinear dimensionality reduction algorithm works by first building a matrix with all pairwise distances between data points. Then, it calculates a new matrix called Laplacian which contains important information on the geometry of the low-dimensional manifold where the data points are assumed to live. The axis for the new data representations are then obtained from the spectral decomposition of the Laplacian matrix (see [32] for more details on the algorithm and [33] for an application on multivariate time series).

Figure 1 illustrates the distribution of data points after each step of RPA applied to the source and target datasets. In this example, we consider that we know the labels of all matrices from the target dataset, i.e., \(\mathcal{T} = \mathcal{T}_t\). Figure 1A shows the point clouds of each dataset, which are clearly unmatched. After recentering (Fig. 1B), stretching (Fig. 1C) and rotating (Fig. 1D), the statistical distributions get matched and the same classifier can be used on both datasets.

C. Classification accuracy after RPA

We compare the classification accuracy results on the simulated dataset for six different methods of Transfer Learning. In each of them, the training and testing datasets are different but the classifier is always the MDM:

- **direct (DCT):** direct use of the points from the source dataset to do classification on the target dataset (no transformation whatsoever).
- **recentering (RCT):** transfer learning considering only the data points of each dataset recentered to \(I_n\). This corresponds to step (1) in the RPA procedure and is similar to what has been done in [9].
- **parallel transport (PRL):** transfer learning using the method proposed in [10]. The procedure is analogous to RCT, but with the points being re-centered to the halfway point along the geodesic path linking the geometric means of each dataset instead of the Identity matrix.
- **optimal transport (OPT):** transfer learning using the optimal transport approach proposed in [7] and adapted to take into account the fact that we have data points defined in the SPD manifold instead of Euclidean vectors.
- **RPA:** transfer learning with matrices transformed using RPA, as described in Section IV-B.
- **calibration:** classification using only the labeled trials available in the target dataset, with no help from the data in the source dataset.

We assess the performance of each method via a randomized cross-validation procedure consisting of:

1) Select \(2n\) random elements from \(\mathcal{T}\) (\(n\) from each class). These data points define \(\mathcal{T}_n\)
2) Define the test dataset \(\mathcal{T}_n\) containing the other \(200 - 2n\) elements of \(\mathcal{T}\)
3) Obtain the accuracy of the classification via MDM for this particular partition of \(\mathcal{T}\)
4) Repeat the three preceeding steps 10 times and get the mean accuracy for each method.

The results in Fig. 2 show that the DCT pipeline gives classification results at chance level (0.5) independently of the number of matrices available in \(\mathcal{T}_t\). We also observe that simply using RCT already greatly improves classification accuracy, as reported in [9]. Our RPA method further improves the results. We also observe that PRL has virtually the same performance as RCT, which is not surprising, since they are both unsupervised methods based on the idea of re-centering the datasets to a common point in the SPD manifold. The results with OPT are equivalent to RCT and PRL as well. The accuracy with calibration improves when the number of available labels in the target dataset increases, eventually converging to the same performance as RPA. This result is not surprising, because with a sufficient amount of data in \(\mathcal{T}_t\) it is already possible to train a good classifier without the need of doing transfer learning.

Our observations in this session are in accordance with the theoretical results of [34], which says that “if there is enough target data, then no source data are needed (...). This is because the possible reduction in error due to additional source data is always less than the increase in error caused by the source data being too far from the target data”. Such a result points to the existence of a certain “saturation” effect in the quality of transfer learning when too many trials are available in the target session, a behavior that could be exploited to decide when to stop transferring information from previous experimental sessions.

VI. NUMERICAL ILLUSTRATIONS: REAL DATA

The analysis on real data were all done using EEG recordings from experiments with Brain-Computer Interfaces (BCI)
experiments. As usual, the EEG signal is a \( n \)-dimensional multivariate time-series and is denoted by \( x(t) \) where each dimension represents an electrode. Each experimental trial \( i \) lasts a few seconds and is associated to a matrix \( X_i \in \mathbb{R}^{n \times T} \), where \( T \) is the number of time samples defining the trial. To every trial we associate a SPD matrix \( C_i \) describing its multivariate statistics and a label \( y_i \) indicating what was the task performed during the trial. The dataset for each subject is composed of a set of couples \( (C_i, y_i) \). Our investigation focuses on the classification accuracy of a MDM classifier that is trained with the data from a source subject and is used to classify the signals from a target subject. We compare the performance of such classifier using the different Transfer Learning strategies described in Section V-C.

### A. The datasets

The investigations were carried out on eight datasets covering three different BCI paradigms. All Motor Imagery datasets are publicly available and were downloaded and pre-processed using the MOABb framework [35]. The P300 dataset comes from experiments performed in our laboratory on the P300-based game Brain Invaders [36]. The SSVEP dataset was the same as the one presented in [37]. See Table I for a brief overview of each dataset’s features.

We estimated the SPD matrices for each BCI paradigm differently. For the MI datasets, the SPD matrices were the spatial covariance matrices of the multivariate EEG recordings. The signals of each trial in the SSVEP paradigm were filtered using bandpass filters around certain frequencies of interest and its SPD matrices were diagonal blocks concatenating the spatial covariance matrices of the filtered signals [1]. For the P300, the SPD matrix of each trial was obtained using the approach from [44], where one estimates a special form of covariance matrix that captures the influence of event-related potentials in each trial.

### B. Comparing cross-subject classification accuracies

In this section, we compare the accuracy of the classification of trials for each pair of target and source subject. The classification is done using the MDM classifier and the values of the accuracies are assessed using the same cross-validation scheme explained in Section V-C.

We begin with a qualitative analysis of the cross-subject classification accuracy using a tool from combinatorial data analysis called seriation [45]. This procedure sorts the lines and columns of a data matrix in order to make relevant patterns appear. In our case, the matrix \( S \) to be re-ordered contains at its \((i, j)\) coordinate the accuracy of the classification using subject \( i \) as target and subject \( j \) as source. The rows of \( S \) are then sorted in decreasing order of columns sum, and the columns of this row-sorted-matrix \( S \) are rearranged in decreasing order of their rows sum. The output of this procedure is a new representation where the pairs of source-target subjects with the best accuracy are located at the top-left region of the matrix, while the worst pairs are at the bottom-right region. Figure 3 shows the results of this seriation procedure on the PhysionetMI dataset for two sizes of \( T_i \) (the number of labeled

---

### Table I

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Paradigm</th>
<th>Subjects</th>
<th>Classes</th>
<th>Trials per class</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhysionetMI</td>
<td>MI</td>
<td>109</td>
<td>2</td>
<td>22</td>
<td>[38]</td>
</tr>
<tr>
<td>Cho2017</td>
<td>MI</td>
<td>50</td>
<td>2</td>
<td>100</td>
<td>[39]</td>
</tr>
<tr>
<td>SSVEP</td>
<td>SSVEP</td>
<td>12</td>
<td>3</td>
<td>8</td>
<td>[37]</td>
</tr>
<tr>
<td>P300</td>
<td>P300</td>
<td>24</td>
<td>2</td>
<td>72 and 360</td>
<td>[36]</td>
</tr>
<tr>
<td>BNCI2014001</td>
<td>MI</td>
<td>9</td>
<td>4</td>
<td>72</td>
<td>[40]</td>
</tr>
<tr>
<td>BNCI2014002</td>
<td>MI</td>
<td>15</td>
<td>2</td>
<td>80</td>
<td>[41]</td>
</tr>
<tr>
<td>BNCI2015001</td>
<td>MI</td>
<td>13</td>
<td>2</td>
<td>100</td>
<td>[42]</td>
</tr>
<tr>
<td>MunichMI</td>
<td>MI</td>
<td>11</td>
<td>2</td>
<td>150</td>
<td>[43]</td>
</tr>
</tbody>
</table>
target trials) and three different pipelines: DCT, RCT, and RPA. We observe that with RCT and RPA there are more pairs of subjects with high values of cross-subject classification than with DCT. In particular, for RCT and RPA we note that there are a few target subjects that have very good accuracy on classification for almost all possible source subjects, a feature that is possibly related to the performance of each target subject to classify its own trials (intra-subject accuracy). This can be interpreted as: subjects that are “good” for classifying their own data should be “good” for receiving information from other source subjects. We also observe a clear improvement in the average value of the cross-subject classification accuracies when more points are available in $T_t$.

Our next analysis consisted in calculating the mean over all cross-subject AUC (Area Under the ROC Curve) for each Transfer Learning pipeline on each dataset. We used these values as quantitative measures for assessing whether one pipeline is better than the other on cross-subject classification. The scores are shown in Table II. We should mention that only the subjects whose intra-score AUC (i.e., classification of its own data) was above chance level were used in these calculations.

Figure 4 shows the results of statistical tests performed on each pair of methods, allowing for a more substantiated assessment of the performance of the methods. The statistical tests comparing method A versus method B were carried out in the following way: (1) For each source subject $j$, we perform a signed paired t-test comparing the scores of method A to method B along all target subjects. Each of these tests yield a statistic $T_j$ and a p-value $p_j$ is obtained via permutations tests [46]. (2) We combine the p-values of all the source subjects using Stouffer’s Z-score method [47]. This yields a single p-value for the comparison between methods as well as the direction to which the null hypothesis has been rejected (i.e., whether method A is better than B or vice-versa). (3) We adjust p-values of each pairwise comparison using Holm’s step-down procedure [48] to account for the multiple comparison problem.

The results in Figure 4 indicate that when there are enough points in $T_t$ (“enough” depending on each dataset), transforming the data points with RCT, PRL or RPA is always better than not doing any distribution matching (DCT). We also observe that most of the time there is no statistical significance between the results with PRL and RCT, as expected, since they both amount to re-centering the datasets to a new point in the SPD manifold. For increasing values of $N$ (the number of labeled trials in the target dataset), RPA gets better in comparison to almost all other methods, as expected and observed in Figure 2 for simulated data. Interestingly, OPT has very poor results in comparison to all other methods, probably because it does not use any prior hypothesis on the statistical distributions of the datasets and has to solve a difficult optimization problem to determine its transportation plan. Lastly, during our statistical analysis of the results, we have observed that better results on Transfer Learning via RPA are often associated to good intra-subject accuracy, since in this case the estimation of the class means is more stable and thus the rotation matrix $U$ is better estimated. This explains why for some databases (e.g. PhysionetMI) the RPA is not necessarily the best method for Transfer Learning and an unsupervised approach like RCT has better results.

C. The role of the size of $T_t$

As pointed out in the simulation results from Section V-C, when the size of $T_t$ increases, using Transfer Learning is no longer relevant, since one may already have enough data to build a good classifier for the target subject. To investigate this behavior on our real datasets, we compared the cross-subject classification accuracy of RPA to that of the calibration method (see Section V-C for details).

Figure 5 shows a scatter plot comparing the classification accuracies on the MunichMI dataset. We see that when $T_t$ grows, there are more pairs of subjects for which the classification using the calibration method on the target subject is better than doing transfer learning via RPA (28% to 34% of all the pairs of subjects). However, the location of the cloud of points in the figure indicates that the Transfer Learning with RPA is still superior to the Calibration method for most pairs of subjects. We used a one-sided paired t-test with random data permutation [46] to compare the accuracies of RPA and calibration on each dataset for different sizes of the $T_t$. The null hypothesis of equivalency between the two methods was rejected ($p < 0.01$) on almost all tests, the only exception being for those on the BCNI2014001 dataset. Moreover, for
Table II: Mean values of the cross-subject AUC (Area Under the ROC Curve) for five pipelines (all described in Section V-C) on eight different datasets. Parameter \( N \) is the number of training points available on the target dataset on each situation. The best method in each instance is written in bold.

<table>
<thead>
<tr>
<th>Pipeline/MI</th>
<th>MEAN AUC</th>
<th>MEAN AUC</th>
<th>MEAN AUC</th>
<th>MEAN AUC</th>
<th>MEAN AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>MunichMI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>DCT</td>
<td>RCT</td>
<td>PRL</td>
<td>OPT</td>
<td>RPA</td>
</tr>
<tr>
<td>1</td>
<td>0.54</td>
<td>0.61</td>
<td>0.65</td>
<td>0.59</td>
<td>0.56</td>
</tr>
<tr>
<td>5</td>
<td>0.55</td>
<td>0.65</td>
<td>0.65</td>
<td>0.60</td>
<td>0.63</td>
</tr>
<tr>
<td>10</td>
<td>0.56</td>
<td>0.67</td>
<td>0.67</td>
<td>0.66</td>
<td>0.66</td>
</tr>
<tr>
<td>15</td>
<td>0.57</td>
<td>0.68</td>
<td>0.68</td>
<td>0.60</td>
<td>0.67</td>
</tr>
<tr>
<td>PhysionetMI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>DCT</td>
<td>RCT</td>
<td>PRL</td>
<td>OPT</td>
<td>RPA</td>
</tr>
<tr>
<td>1</td>
<td>0.54</td>
<td>0.61</td>
<td>0.65</td>
<td>0.59</td>
<td>0.57</td>
</tr>
<tr>
<td>5</td>
<td>0.55</td>
<td>0.61</td>
<td>0.61</td>
<td>0.57</td>
<td>0.59</td>
</tr>
<tr>
<td>10</td>
<td>0.55</td>
<td>0.62</td>
<td>0.62</td>
<td>0.57</td>
<td>0.62</td>
</tr>
<tr>
<td>15</td>
<td>0.57</td>
<td>0.64</td>
<td>0.64</td>
<td>0.58</td>
<td>0.66</td>
</tr>
<tr>
<td>P300-Erwan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>DCT</td>
<td>RCT</td>
<td>PRL</td>
<td>OPT</td>
<td>RPA</td>
</tr>
<tr>
<td>1</td>
<td>0.58</td>
<td>0.69</td>
<td>0.65</td>
<td>0.62</td>
<td>0.62</td>
</tr>
<tr>
<td>5</td>
<td>0.59</td>
<td>0.73</td>
<td>0.73</td>
<td>0.65</td>
<td>0.71</td>
</tr>
<tr>
<td>10</td>
<td>0.61</td>
<td>0.76</td>
<td>0.76</td>
<td>0.65</td>
<td>0.76</td>
</tr>
<tr>
<td>15</td>
<td>0.64</td>
<td>0.78</td>
<td>0.77</td>
<td>0.66</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Fig. 4. Results of the statistical tests on each pair of pipelines for all possible values of \( N \) on each dataset as indicated in Table II (for instance, on Cho2017 we have \( N_1 = 1, N_2 = 5, N_3 = 10, \) and \( N_4 = 25 \)). The color/pattern of the squares indicate whether there’s no statistical difference between two methods (white squares), if the Left method is superior to the Right one (L and R in the legend) (dark gray squares) or the contrary (squares with crossed patterns). All conclusions are with \( p < 0.05 \) corrected via Holm’s adjustment [48]. For instance, we see that for dataset Cho2017, the method RPA is inferior to RCT when \( N = 1 \), but RPA becomes superior when \( N = 25 \). Furthermore, for this same dataset, there’s no statistical difference in the comparison between PRL and RCT for any \( N \).

Fig. 5. Scatter plots comparing the accuracies of the cross-subject classification on the MunichMI dataset for the RPA and Calibration methods. We consider two sizes for \( T_r \). The percentage numbers indicate the proportion of dots above or below the diagonal line.

the tests where \( H_0 \) was rejected, we observed a superiority of RPA in comparison to calibration.

D. Combining information from multiple subjects

We investigated how the matching of statistical distributions via RPA affects the performance of two baseline methods for gathering information from the data of multiple subjects: pooling and ensembling. The classification for each target subject is done using information coming from all other source subjects available in the database. Following the same approach as in previous sections, we only considered source subjects featuring an intra-subject accuracy above chance level, i.e., subjects in which it is meaningful to use transfer learning. The experiments were done on the PhysionetMI dataset with \( |T_r| = 15 \) labeled trials available for each target subject and the Cho2017 dataset with \( |T_r| = 25 \).

The pooling strategy consists of gathering for each target subject the data from all other source subjects into one big dataset. Then, a classifier is trained on the pooled dataset and used to infer the trials from the target subject. We compared the performance of a MDM classifier when the source subjects were pooled with no transformation (DCT) to when the statistical distributions of each source subject were matched to that from the target subject using RCT or RPA (PRL is not fit for pooling, since the matrices are not all centered to the same place in the SPD manifold). The boxplots in Figure 6 show the distributions of the classification scores of each of the target subjects. Using pairwise one-sided paired \( t \)-test with random permutations, the null hypothesis of equivalency between the scores of DCT, RCT, and RPA were all rejected
The pooling results show a clear improvement in the average score for the source RPA at least 15% between datasets. For the Cho2017 ensembling and pooling strategies for different methods of statistical matching Fig. 6. Box plots with the distribution of the classification scores for the IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING, VOL. NO. 11 indicate that the ensembling strategy with RPA for multiple comparisons). The results of the statistical tests compared to DCT in the PhysionetMI dataset (p < 0.05) but they are equivalent for the Cho2017 dataset (p = 0.23). The RCT method is superior to DCT for both datasets (p < 0.01) whereas the scores with PRL are equivalent to DCT for both datasets. We see then that the ensembling strategy can also be improved when adding an extra step for matching the statistics of the datasets of each pair of source-target subjects.

VII. CONCLUSION AND FUTURE WORKS

In this paper, we have presented a new method for overcoming the negative effects of statistical distribution mismatch between subjects in BCI classification. Our proposal consists of a sequence of geometrical transformations on the data points from two sessions with the intention of making the shapes of their point clouds in a high-dimensional space as similar as possible. The inspiration for this method comes from Procrustes Analysis, however, here the method has been adapted to the case where the data points are SPD matrices and live in a Riemannian manifold, which is here proposed for the first time. Another relevant theoretical contribution is the mathematical framework proposed in Section IV-B, which includes the methods in [9] and [10] and extends them, leading to our RPA method. Such formalism allows for a better understanding of the intrinsic assumptions regarding the statistics of the data points during the distribution matching procedure.

An important aspect of our proposal is that we exploit the availability of supervised information in the source session as well as the sequential nature of the trials in the target session. It should be noted, however, that when no labels are available for the target session, a re-centering of data points based solely on the geometric means of each dataset (which does not rely on any supervised information) already greatly improves the cross-session and cross-subject classification, as first noted in [9]. This would be the case, for instance, in BCI applications for people with extreme motor disability, where the labeling of classes is very challenging. In this kind of situation, one may still perform the re-centering and stretching steps of the RPA method for matching the statistical distributions, turning the transfer learning procedure into a completely unsupervised one.

We have assessed the superiority of the RPA method on several publicly available datasets and have used a heterogeneous panel of statistical tools to analyze the results. Also, we have included in our study other recent contributions from the literature, leading to a comprehensive comparison of the performance of state-of-the-art methods. We hope that the breadth of the analysis performed here will be useful as a reference for future works related to Transfer Learning on the SPD manifold. In order to foster reproducible research, complete Python code for the results in this paper is available at https://github.com/plcrodrigues/RPA.

Future perspectives for this work shall include an online implementation of our method, where usual drifts in statistics from signals on the same session would be corrected via distribution matching. An important challenge for such procedure would be to detect when changes in the statistics occur as well as when the number of new trials is already large enough so that no information from data points drawn from previous statistical distributions are needed. Another interesting line of work would be to go further in the analysis of Section VI-D by extending the methods proposed in [12], [14], and [13] with a statistical matching step based on the RPA. Finally, another interesting topic to investigate would be to consider a Transfer Learning approach for matching data from subjects having different numbers of electrodes.

VIII. ACKNOWLEDGEMENT

This work is partly supported by the ERC Grant CHESS 2012-ERC-AdG-320684. We thank Florent Bouchard for his help on the optimization procedure in Equation (35).