Averaging trajectories on the manifold of symmetric positive definite matrices

Thibault de Surrel, Florian Yger
LAMSADE, CNRS, PSL Univ. Paris-Dauphine
Paris, France
thibault.de-surrel@lamsade.dauphine.fr
florian.yger@lamsade.dauphine.fr

Sylvain Chevallier
TAU, LISN, University Paris-Saclay
Gif-sur-Yvette, France
sylvain.chevallier@universite-paris-saclay.fr

Fabien Lotte
Inria center at the
University of Bordeaux / LaBRI
Talence, France
fabien.lotte@inria.fr

***Abstract*—The goal of this paper is to leverage more information from a single measurement (e.g. an ElectroEncephaloGraphic (EEG) trial) by representing it as a trajectory of covariance matrices (indexed by time for example) instead of a single aggregated one. Doing so, we aim at reducing the impact of non-stationarities and variabilities (e.g. due to fatigue or stress for EEG). Covariance matrices being symmetric positive definite (SPD) matrices, we present two algorithms to classify trajectories on the space of SPD matrices. These algorithms consist in computing, in two different ways, the mean trajectory of a set of training trajectories and use them as class prototypes. The first method computes a pointwise mean and the second one achieves a smart matching using the Dynamic Time Warping (DTW) algorithm. As we are considering SPD matrices, the geometry used along these processes is the Riemannian geometry of the SPD matrices. We tested our algorithms on synthetic data and on EEG data from six different datasets. We show that our algorithms yield better average results than the state-of-the-art classifier for EEG data.**

***Index Terms*—Brain-computer interfaces, Covariance matrices, Electroencephalography, Riemannian geometry, Time series analysis**

I. Introduction

Covariance matrices have achieved great successes in many scientific areas such as Brain-Computer Interfaces (BCIs) [1], process control [2] or biomedical image analysis [3]. In this paper, we will focus on applications for BCIs where the goal is to translate brain signals into commands. Non-invasive BCIs mainly use ElectroEncephaloGraphic (EEG) signals recorded using a cap equipped with multiple sensors [4]. The recorded EEG takes the form of a multivariate time series. One can compute the covariance matrix of this signal to better understand the link between the different sensors. The goal is to detect and classify specific patterns in the EEG and to link them to specific commands. In a Motor Imagery (MI) paradigm [5], the subject is, for example, asked to imagine moving his right or left arm and the goal is to discriminate EEG patterns corresponding to those two different classes.

Covariance matrices have a special structure: they are symmetric, positive definite (SPD). Thus, the natural geometry to manipulate them is the affine invariant Riemannian geometry [6]. This geometry leads to a curved space, where the shortest path between two SPD matrices is not a straight line, but rather a geodesic. It has been shown that using this Riemannian geometry leads to state-of-the-art EEG classification performances [1], [7]. The seminal work [8] introduces an algorithm called *Minimum Distance to Mean* (MDM) that uses the tools of Riemannian geometry on SPD matrices in order to classify SPD matrices and therefore, covariance matrices of EEG signals. To achieve this, a mean SPD matrix is first estimated for each class from the training data. Then, given a new SPD matrix, the predicted class is the one for which the mean matrix is the closest to the given SPD matrix, where the distance is the Riemannian distance on the SPD manifold.

Our goal is to extend this approach. In fact, it is well known that EEG recordings are subject to numerous variabilities [9], [10] from the environmental conditions, the subjects' cognitive states (interdays or intersubject variabilities), their fatigue or the task requirements. The representation using covariance matrices might fail to capture those variabilities as they are not able to capture neither temporal dynamics nor frequency information. Therefore, we propose in our work to extend the MDM algorithm [8] by not only using one SPD matrix per EEG trial (its covariance), but several SPD matrices per trial, forming a trajectory (e.g. covariance matrices across time). We then compute a mean trajectory per class instead of a single mean matrix. This way, we hope to leverage more information out of a single EEG trial and therefore, better tackle the variabilities and the non-stationarities. We propose two methods to compute the mean trajectory, on the one hand using a point-wise mean, and on the other hand based on an optimal matching computed using the Dynamic Time Warping (DTW) algorithm. Considering trajectories of covariance matrices has already been proposed to classify brain signals. In [11], they build their trajectories using the discrete Fourier transform and learn an optimal distance between trajectories based on a weighting matrix to classify EEGs. In [12], they build trajectories by estimating a covariance matrix on a sliding window then build a distance on trajectories of SPD matrices. They also propose a dimension reduction algorithm to facilitate the computations and apply it to functional MRI. In [13], the considered trajectories on a Riemannian manifold are geodesics that derive from an unknown group-average trajectory. None of these previous

This work was funded by the French National Research Agency for project PROTEUS (grant ANR-22-CE33-0015-01).

works try to extend the MDM algorithm by computing a mean trajectory and comparing a new sample to this mean trajectory.

The paper is organized as follows: in Section II, after some reminders on the Riemannian geometry of SPD matrices and on the DTW algorithm, we present our two algorithms: *PT-MDM* (for Pointwise Trajectory-MDM) and *DTW-MDM*. Theses methods are tested on synthetic datasets and on real BCI datasets in Section III.

II. Proposed Methods

A. The Riemannian geometry of SPD matrices

We consider the set $\mathcal{P}_c$ of *symmetric, positive definite* (SPD) matrices of size $c \times c$ defined as follows:

$$\mathcal{P}_c = \{P \in \mathbb{R}^{c \times c} \ | P^\top = P, \forall x \in \mathbb{R}^c \text{ s.t. } x \neq 0, \ x^\top P x > 0\}$$

This set can be seen as a Riemannian manifold of dimension $c(c+1)/2$. We can define a distance between two SPD matrices. In this paper, we use the *affine-invariant metric* [14]:

$$\delta_R(P_1, P_2) = \| \log(P_1^{-1/2} P_2 P_1^{-1/2}) \|_F \qquad (1)$$

where $\| \|_F$ is the Frobenius norm and $\log$ the matrix logarithm.

Provided with n SPD matrices $P_1, ..., P_n \in \mathcal{P}_c$, one may need to compute the mean of these matrices. The *Riemannian mean* [15] is defined as follows:

$$\mathfrak{G}(P_1, ..., P_n) = \underset{P \in \mathcal{P}_c}{\text{argmin}} \sum_{i=1}^{n} \delta_R^2(P, P_i) \qquad (2)$$

The Riemannian mean exists and is unique in the case of a manifold of non-positive sectional curvature [16] (such as the manifold of SPD matrices) however, there is no closed-form expression of it. One can use a Riemannian gradient descent algorithm to find an approximate solution [6]. We used Pymanopt [17] to solve such Riemannian optimization problems in Python.

B. Dynamic Time Warping

Dynamic Time Warping (DTW) [18] is a well-known algorithm used to find the optimal alignment between two time series. Let $X = (x_1, ..., x_N)$ and $Y = (y_1, ..., y_M)$ be two sequences of size respectively N and M and let $\mathcal{L}$ be a cost function. The DTW algorithm computes a path $P = ((i_1, j_1), ..., (i_{K_p}, j_{K_P})) \in (\mathbb{N} \times \mathbb{N})^{K_P}$, $K_P \in \mathbb{N}$ between the elements of X and of Y that minimizes the sum of cost:

$$w(P) = \sum_{k=1}^{K_P} \mathcal{L}(x_{i_k}, y_{j_k}).$$

An acceptable path $P = ((i_1, j_1), ..., (i_{K_P}, j_{K_P})) \in (\mathbb{N} \times \mathbb{N})^{K_P}$, $K_P \in \mathbb{N}$ is a sequence that is continuous ($i_k - i_{k-1} \leq 1$ and $j_k - j_{k-1} \leq 1$), monotonic ($i_{k-1} \leq i_k$ and $j_{k-1} \leq j_k$) and bounded ($(i_1, j_1) = (1, 1)$ and $(i_{K_P}, j_{K_P}) = (N, M)$). The final DTW distance is computed as follows:

$$\text{DTW}(X, Y) = \sqrt{\min_{P \in \Pi} w(P)}$$

where Π is the set of acceptable paths. The minimizing path can be computed in $O(NM)$ operations using dynamic programming [19]. A linear in time and space algorithm called FastDTW [20] have been developed to approach the DTW.

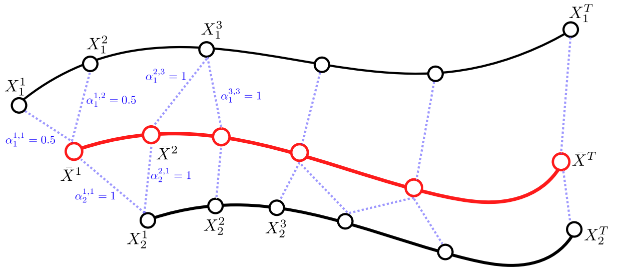

Fig. 1. Example of a matching using the *DTW-MDM* method. Training trajectories $\{X_i^t\}_{i=1,...,N}^{t=1,...,T}$ are in black and the mean trajectory $\bar{X}$ is in red. The blue dashed lines represent the matching computed by the DTW algorithm. Equation 3 is then used to compute the weights $\alpha_i^{t,t'}$.

C. The two proposed algorithms

In this work, we want to consider trajectories of SPD matrices, such as trajectories of covariance matrices indexed by time or frequency for example. Let us consider N training trajectories of SPD matrices of length T denoted $\{X_i^t\}_{i=1,...,N}^{t=1,...,T}$ where $X_i^t \in \mathcal{P}_c$ is the t-th matrix of the i-th trajectory. Our goal is to find the mean trajectory $\bar{X} = \{\bar{X}^t\}^{t=1,...,T}$. We propose two different methods to do this[1].

First method: PT-MDM: The first method is called *PT-MDM* for *Pointwise Trajectory-MDM*. It is the natural way of thinking of a mean trajectory: each point of the mean trajectory $\bar{X}$ is the pointwise Riemannian mean (see Eq. 2) of the corresponding points of the training trajectories $\{X_i\}_{i=1,...,N}$:

$$\forall t \in \{1, ..., T\}, \ \bar{X}^t = \underset{X \in \mathcal{P}_c}{\text{argmin}} \sum_{i=1}^{N} \delta_R^2(X, X_i^t).$$

Second method: DTW-MDM: The second method is called *DTW-MDM* and uses the DTW algorithm to align trajectories in order to take into account possible variabilities such as time shifts, dilatation or contraction of time. This algorithm is iterative and, after randomly generating the initial mean trajectory $\bar{X}$, it iterates two steps until convergence:

1) A matching is computed between each of the training trajectories $\{X_i^t\}^{t=1,...,T}$ and the current mean trajectory $\bar{X}$. This step gives a set of coefficients $\{\alpha_i^{t,t'}\}_{i=1,...,N}^{t,t'=1,...,T}$ where the coefficient $\alpha_i^{t,t'}$ represents the influence of $X_i^{t'}$ on $\bar{X}^t$.
2) A weighted Riemannian mean is computed:

$$\bar{X}^t = \underset{X \in \mathcal{P}_c}{\text{argmin}} \sum_{i=1}^{N} \sum_{t'=1}^{T} \alpha_i^{t,t'} \delta_R^2(X, X_i^{t'})$$

To find the coefficients $\{\alpha_i^{t,t'}\}_{i=1,...,N}^{t,t'=1,...,T}$, we use the DTW, presented in II-B, with the cost $\mathcal{L}$ being the squared Riemannian distance δ_R^2. Indeed, for all $i \in \{1, ..., N\}$, the DTW algorithm gives a path $P_i = ((t'_1, t_1), ..., (t'_{K_P}, t_{K_P}))$ that matches $\{X_i^{t'}\}^{t'=1,...,T}$ to $\bar{X} = \{\bar{X}^t\}^{t=1,...,T}$. Using this path, one can construct the weights $\{\alpha_i^{t,t'}\}^{t,t'=1,...,T}$:

[1] Find our code at https://github.com/thibaultdesurrel/Trajectory-MDM

$$\alpha_i^{t,t'} = \begin{cases} \dfrac{1}{|\{\tilde{t}' \ : \ (\tilde{t}', t) \in P_i\}|} & \text{if } (t', t) \in P_i \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where $|A|$ is the cardinality of the set A. The coefficient $\alpha_i^{t,t'}$ is therefore the inverse of the number of points that are linked to $\bar{X}^t$ by the DTW. We show an example in Figure 1. An important feature of our method is, since the DTW algorithm can process time series of varying lengths, we have the possibility of having less points on the mean trajectory $\bar{X}$ than on the training trajectories. This can possibly help to summarize the information and reduce the noise of the training trajectories. Two criteria are used to check convergence: a maximum number of iterations is given as well as a threshold for the norm of the difference between two consecutive mean trajectories. For all of our experiments, we used a maximum number of 10 iterations and a threshold of 10^{-5}.

Classification: For training a classifier using either these methods, we start by computing the mean trajectory $\bar{X}_k$ for each class k using either the pointwise or DTW method. Once the mean trajectories are computed, for classifying a new trajectory X, the distances (computed pointwise or using the DTW based on the method used to compute the mean trajectories) between X and the mean trajectory of each class $\bar{X}_k$ is computed. The class $\hat{k}$ for which the distance is the minimum is returned.

III. Numerical Experiments

After describing the algorithms that we propose, we can test them. We now show some experiments, first on synthetic datasets and then on real data.

A. Synthetic experiments

1) Data generation: To build a synthetic dataset we start by sampling two matrices M_1 and M_N using the spectral decomposition: $M_1 = U_1^T D_1 U_1$ (resp. $M_N = U_N^T D_N U_N$) where the diagonal matrix $D_1 \in \mathbb{R}^{c \times c}$ (resp D_N) has strictly positive values drawn from a uniform distribution $\mathcal{U}([0, 5])$ and where the orthogonal matrix U_1 (resp. U_N) is drawn from the $O(c)$ Haar distribution (the only uniform distribution on $O(c)$) [21]. These two matrices are the beginning and end points of the underlying trajectory. We can then sample $M_2, ..., M_{N-1}$ uniformly along the geodesic going from M_1 to M_N and add some Gaussian noise to them (we sample $\mu_i \sim \mathcal{N}(0, \frac{1}{2} I_c)$ and add $\mu_i \mu_i^T$ to M_i). This will give us the "true" trajectory for the first class. For the second class, we simply modify one matrix M_i among the first underlying trajectory $M_2, ...M_{N-1}$ by adding another SPD matrix sampled like M_1 and M_N (with eigenvalues drawn uniformly in [0,1]). We get a new matrix $\tilde{M}_i$ and the underlying trajectory for the second class is $(M_1, ..., M_{i-1}, \tilde{M}_i, M_{i+1}, ..., M_N)$. At this step, we have the underlying trajectory of both classes. We then wish to mimic the randomness that occurs while measuring real life data. For each class, we sample n trajectories of l points that follows the corresponding underlying trajectory. To do this, for a given trajectory, we start by sampling uniformly l times $t_1, ..., t_l$ in $[0, 1]$. This gives us the times at which we "recorded" a new point on the trajectory. Then the j^{th} point on the trajectory is sampled on the geodesic between $M_{\frac{i}{N}}$ and $M_{\frac{i+1}{N}}$ where i is such that $\frac{i}{N} \leq t_j < \frac{i+1}{N}$. Finally, we add a Gaussian noise (with the same parameters as above) to all the samples points.

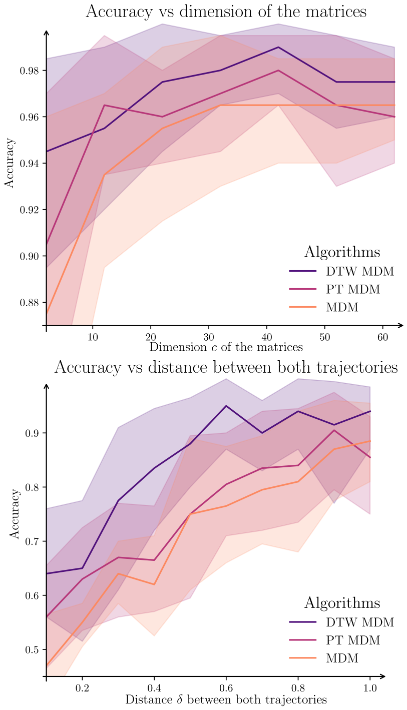

Fig. 2. Results on the synthetic datasets.

Dataset	Number of subjects	Number of channels	Number of trials	Sampling frequency	Trial length
BNCI2014001 [22]	9	22	144	250 Hz	3 s
BNCI2014002 [23]	15	15	80	512 Hz	5 s
BNCI2014004 [24]	9	3	360	250 Hz	4.5 s
BNCI2015001 [25]	12	13	200	512 Hz	5 s
Zhou2016 [26]	4	14	160	250 Hz	5 s
AlexMI [27]	8	16	20	512 Hz	3 s

TABLE I
Summary of the datasets considered during the study

2) Influence of the parameters: In this experiment we compare the two algorithms *PT-MDM* and *DTW-MDM* presented in Section II with each other. We also compare them with a classical MDM algorithm, where we summarize all the information of a trajectory to its Riemannian mean and compute a unique SPD mean matrix for each class. The Riemannian distance is then used to classify a new matrix. We want to see if considering trajectories is better than considering a single SPD matrix, and to compare a smart matching using the DTW to a trivial one (*DTW-MDM vs PT-MDM*). The parameters investigated are: the number l of points on the trajectories, the dimension c of the matrices, the additive Gaussian noise ε and the distance between the two classes δ. The base parameters are: $l = 10$, $c = 2$, $\varepsilon = \frac{1}{2}$. We chose $N = 5$ matrices on the underlying trajectories throughout the experiments.

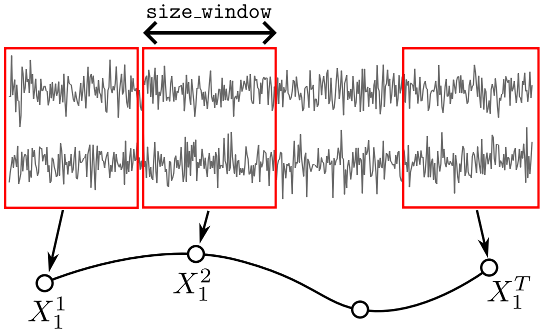

Fig. 3. From a EEG to a trajectory. Each point on the trajectory is a covariance matrix computed on a sub-window of size `size_window` of the EEG.

We show the results for the parameter c and δ in Figure 2. Each point is the mean accuracy of a classifier over 5 simulations. We can see that the two proposed algorithms are better than the classical MDM almost all of the time and that the *DTW-MDM* is always better than *PT-DTW*. It is also the case for the experiments for the parameters l and ε that are not shown because of page limitation. Therefore, we can say that having a matching computed using the DTW algorithm is worth it, and actually performs better than a simple MDM or than a pointwise matching. We also did some experiments where the distances used where the Euclidean one and the results (not shown here) revealed that is was better to consider the Riemannian metric of Eq. 1 to compare SPD matrices.

B. Experiments on BCI dataset

We conducted some experiments on real BCI datasets using MOABB [28]. We selected 6 different motor imagery datasets consisting of several subjects for each dataset and several sessions for each subject, all with balanced classes. A summary of all the datasets is given in Table I. We start by applying a standard band-pass filter with range $[7; 35]$ Hz for each dataset. To compute the covariance matrices, we used the Ledoit-Wolf shrunk covariance matrix [29] to avoid having problems with ill-conditioned matrices.

a) Creating a trajectory given an EEG: To create a trajectory of SPD matrices from an EEG trial, we cut the EEG trial into smaller windows of size `size_window` (a hyperparameter) and compute a covariance matrix for each window. Therefore, we have several SPD matrices for each EEG trial. This procedure is illustrated in Figure 3. For the *DTW-MDM* algorithm, we also have a second hyperparameter `size_mean_traj` that controls the number of points on $\bar{X}$ because, as said in Section II-C, when using *DTW-MDM*, the mean trajectory can have fewer points than the training trajectories. Once the training trajectories have been computed, we apply a Fisher Geodesic Discriminant Analysis (FGDA) filter [30] to all of the covariance matrices. We have one filter per class that has been fitted on all the covariance matrices of all the trajectories corresponding to each class. We can then compute the mean trajectory $\bar{X}_k$ and classify some new EEG.

b) The results: We conducted experiments both intrasubject, where for each subject, the dataset was split into 80% training and 20% testing, and intersubject, where we trained on every subject except one and tested on the last subject. The results of both methods are given in Table III and are compared to the FgMDM, that is a MDM classifier with a FGDA filter as presented in [30]. The two hyperparameters `size_window` and `size_mean_traj` were optimized for each dataset based on the training data and the accuracy is the mean accuracy over all subjects that was cross-validated over 5 folds. We can see that, on real datasets, both proposed methods are almost always better than the FgMDM that has only one covariance matrix. However, it is not clear whether *PT-MDM* or *DTW-MDM* is overall better on real data, even if *PT-MDM* seams to be slightly better on the tested datasets.

Method	Time to train the classifier	Average time to classify a new EEG
FgMDM	0.60 sec	0.0006 sec
PT-MDM	1.17 sec	0.004 sec
DTW-MDM	61.7 sec	0.007 sec

TABLE II
COMPUTATIONAL TIMES ON DATASET ZHOU2016 [26].

The hyperparameters `size_mean_traj` as well as `size_window` are not consistent throughout all datasets and must be optimized for each one. However, the best sizes for the windows that are extracted from an EEG trial are always around one second (from 0.75 sec to 1.3 sec), and the EEG trial length being between 3 and 5 seconds. This corresponds to 3 to 6 points on the training trajectories. We observed that the *DTW-MDM* classifier works better when there is one or two fewer points on the mean trajectory than on the training trajectories. This could be explained by a smoothing effect, that reduces the noise present in the original data. What we also noted is that, most of the time, the best hyperparameters where the same for a same dataset, whether or not we were doing intersubject or intrasubject classification.

c) Computational time: For the dataset Zhou2016 [26], we give the computational times of the different algorithms in Table II. As expected, one can see that the two proposed methods take longer to train than the usual FgMDM. One can also see that the DTW-MDM is the longest to train. This is not surprising as instead of computing a single mean SPD matrix, our algorithms compute a whole mean trajectory, with a complex algorithm for the DTW-MDM. However, once the classifier is trained, classifying a new EEG is very fast ($\sim 10^{-3}$ seconds), making the proposed algorithms well suited for online real-time classification, see, e.g., [31].

IV. FUTURE WORKS

Future works could try to use a differentiable version of the DTW as introduced in [32] to have a fully differentiable loss. Another track could be to use optimal transport to transport the training trajectories onto the mean trajectory and then using the transport plan to compute the coefficients $\{\alpha_i^{t,t'}\}_{i=1,\dots,N}^{t,t'=1,\dots,T}$. We would also like to understand why, although the *DTW-MDM* seems to perform better on synthetic datasets, it is not always the case on real BCI datasets.

	Intrasubject			Intersubject		
Dataset	Accuracy FgMDM	Accuracy DTW-MDM	Accuracy PT-MDM	Accuracy FgMDM	Accuracy DTW-MDM	Accuracy PT-MDM
BNCI2014001	80.5 % (± 1.2)	81.8 % (± 1.7)	**81.6 % (± 1.6)**	63.4 % (± 0.9)	65.4 % (± 1.3)	**66.9 % (± 1.5)**
BNCI2014002	78.5 % (± 1.6)	79.3 % (± 1.4)	**80.6 % (± 1.6)**	61.1 % (± 1.5)	**62.7 % (± 2.2)**	**62.7 % (± 1.8)**
BNCI2014004	73.8 % (± 1.0)	74.1 % (± 1.3)	**75.3 % (± 1.3)**	65.6 % (± 0.6)	68.9 % (± 0.7)	**69.1 % (± 0.8)**
BNCI2015001	85.5 % (± 0.8)	85.1 % (± 1.1)	**86.5 % (± 1.0)**	56.2 % (± 0.4)	**59.0 % (± 0.7)**	59.0 % (± 0.8)
Zhou2016	86.5 % (± 0.7)	88.4 % (± 0.5)	**89.0 % (± 0.5)**	76.1 % (± 0.8)	77.4 % (± 0.7)	**78.1 % (± 0.6)**
Zhou2016 (3 classes)	82.4 % (± 0.6)	**83.8 % (± 0.3)**	83.7 % (± 0.4)	70.1 % (± 1.1)	**71.5 % (± 0.8)**	71.2 % (± 0.8)
AlexMI	**77.1 % (±3.1)**	76.5 % (± 3.6)	76.2 % (± 3.0)	57.8 % (± 1.3)	58.7 % (± 1.3)	**59.3 % (± 1.9)**

TABLE III
RESULTS ON BCI DATASETS

V. CONCLUSION

In this paper we presented two new approaches to the MDM algorithm to classify EEG: instead of considering a single SPD matrix per EEG trial, we computed a trajectory of SPD matrices indexed by time for each EEG trial. We first tested our algorithms on synthetic datasets, to assess their performances. This study concludes that finding a smart matching and using it in a weighted average was better than just computing a pointwise average. Finally, we tested our algorithm on real BCI datasets, both intrasubjects and intersubjects. In both cases, the proposed algorithms performed better, on average, than the state of the art FgMDM algorithm. In our experiments, we considered trajectories indexed by time, however, the two proposed algorithms can work with any type of trajectories of SPD matrices, no matter how the trajectory is indexed.

REFERENCES

[1] F. Yger, M. Berar, and F. Lotte, “Riemannian Approaches in Brain-Computer Interfaces: A Review,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 10, pp. 1753–1762, Oct. 2017.

[2] M. Willjuice Iruthayarajan and S. Baskar, “Covariance matrix adaptation evolution strategy based design of centralized pid controller,” *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5775–5781, 2010.

[3] X. Pennec, “3 - manifold-valued image processing with spd matrices,” in *Riemannian Geometric Statistics in Medical Image Analysis*, X. Pennec, S. Sommer, and T. Fletcher, Eds. Academic Press, 2020, pp. 75–134.

[4] M. van Gerven, J. Farquhar, R. Schaefer, R. Vlek, J. Geuze, A. Nijholt, N. Ramsey, P. Haselager, L. Vuurpijl, S. Gielen, and P. Desain, “The brain–computer interface cycle,” *Journal of Neural Engineering*, vol. 6, no. 4, p. 041001, jul 2009.

[5] G. Pfurtscheller and C. Neuper, “Motor imagery and direct brain-computer communication,” *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[6] X. Pennec, P. Fillard, and N. Ayache, “A Riemannian Framework for Tensor Computing,” *Int J Comput Vis*, vol. 66, no. 1, pp. 41–66, 2006.

[7] R. Roy, M. Hinss, L. Darmet, S. Ladouce, E. Jahanpour, B. Somon, X. Xu, N. Drougard, F. Dehais, and F. Lotte, “Retrospective on the first passive brain-computer interface competition on cross-session workload estimation,” *Frontiers in Neuroergonomics*, vol. 3, p. 838342, 2022.

[8] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, “Multiclass Brain–Computer Interface Classification by Riemannian Geometry,” *IEEE. Trans. Biomed. Eng.*, vol. 59, no. 4, pp. 920–928, Apr. 2012.

[9] S. Saha and M. Baumert, “Intra- and Inter-subject Variability in EEG-Based Sensorimotor Brain Computer Interface: A Review,” *Front. Comput. Neurosci.*, vol. 13, 2020.

[10] T. Krumpe, K. Baumgaertner, W. Rosenstiel, and M. Spüler, “Non-stationarity and inter-subject variability of eeg characteristics in the context of bci development,” in *Graz BCI Conference*, 2017.

[11] Y. Li, K. Wong, and H. Debruin, “EEG signal classification based on a Riemannian distance measure,” *IEEE Toronto International Conference - Science and Technology for Humanity*, Sep. 2009.

[12] M. Dai, Z. Zhang, and A. Srivastava, “Analyzing Dynamical Brain Functional Connectivity As Trajectories on Space of Covariance Matrices,” *IEEE Trans. Med. Imag.*, vol. 39, no. 3, pp. 611–620, 2020.

[13] J.-B. Schiratti, S. Allassoniere, O. Colloit, and S. Durrelman, “Learning spatiotemporal trajectories from manifold-valued longitudinal data,” in *NeurIPS*, vol. 28. Curran Associates, Inc., 2015.

[14] R. Bhatia, *Positive Definite Matrices*, ser. Princeton Series in Applied Mathematics. Princeton, N.J: Princeton University Press, 2007.

[15] M. Moakher, “A Differential Geometric Approach to the Geometric Mean of Symmetric Positive-Definite Matrices,” *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 3, pp. 735–747, Jan. 2005.

[16] H. Karcher, “Riemannian center of mass and mollifier smoothing,” *Commun Pure Appl Math*, vol. 30, no. 5, pp. 509–541, 1977.

[17] J. Townsend, N. Koep, and S. Weichwald, “Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation,” *JMLR*, vol. 17, no. 137, p. 1–5, 2016.

[18] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series.” in *KDD Workshop*, U. M. Fayyad and R. Uthurusamy, Eds. AAAI Press, 1994, pp. 359–370.

[19] M. Müller, “Dynamic Time Warping,” in *Information Retrieval for Music and Motion*. Berlin, Heidelberg: Springer, 2007, pp. 69–84.

[20] S. Salvador and P. Chan, “FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space,” *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, 2010.

[21] F. Mezzadri, “How to Generate Random Matrices from the Classical Compact Groups,” *Notices of the AMS*, vol. 54, no. 5, 2007.

[22] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K. Miller, G. Mueller-Putz, G. Nolte, G. Pfurtscheller, H. Preissl, G. Schalk, A. Schlögl, C. Vidaurre, S. Waldert, and B. Blankertz, “Review of the bci competition iv,” *Frontiers in Neuroscience*, vol. 6, 2012.

[23] D. Steyrl, R. Scherer, J. Faller, and G. Müller-Putz, “Random forests in non-invasive sensorimotor rhythm brain-computer interfaces: A practical and convenient non-linear classifier,” *Biomedizinische Technik/Biomedical Engineering*, vol. 61, pp. 77–86, 04 2015.

[24] R. Leeb, F. Lee, C. Keinrath, R. Scherer, H. Bischof, and G. Pfurtscheller, “Brain–computer communication: Motivation, aim, and impact of exploring a virtual apartment,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 15, no. 4, pp. 473–482, 2007.

[25] J. Faller, C. Vidaurre, T. Solis-Escalante, C. Neuper, and R. Scherer, “Autocalibration and recurrent adaptation: Towards a plug and play online erd-bci,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, pp. 313–9, 04 2012.

[26] B. Zhou, X. Wu, Z. Lv, L. Zhang, and X. Guo, “A fully automated trial selection method for optimization of motor imagery based brain-computer interface,” *PLOS ONE*, vol. 11, no. 9, pp. 1–20, 2016.

[27] A. Barachant, “Commande robuste d’un effecteur par une interface cerveau machine EEG asynchrone,” Theses, Université de Grenoble, Mar. 2012. [Online]. Available: https://theses.hal.science/tel-01196752

[28] V. Jayaram and A. Barachant, “Moabb: trustworthy algorithm benchmarking for bcis,” *J Neural Eng*, vol. 15, no. 6, p. 066011, 2018.

[29] O. Ledoit and M. Wolf, “A well-conditioned estimator for large-dimensional covariance matrices,” *J Multivar Anal*, vol. 88, no. 2, pp. 365–411, 2004.

[30] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, “Riemannian geometry applied to BCI classification,” in *LVA/ICA 2010*, vol. 6365. Springer, Sep. 2010, p. 629.

[31] C. Benaroch, K. Sadatnejad, A. Roc, A. Appriou, T. Monseigne, S. Pramij, J. Mladenovic, L. Pillette, C. Jeunet, and F. Lotte, “Long-term bci training of a tetraplegic user: Adaptive riemannian classifiers and user training,” *Frontiers in Human Neuroscience*, vol. 15, 2021.

[32] M. Cuturi and M. Blondel, “Soft-DTW: A Differentiable Loss Function for Time-Series,” Feb. 2018.