UCLouvain

École polytechnique de Louvain

Riemannian classification of covariance matrices coming from EEG signals

Author: **Pierre VELDEMAN**
Supervisors: **Pierre-Antoine ABSIL, Estelle MASSART**
Readers: **Noémie JAQUIER, John LEE**
Academic year 2020–2021
Master [120] in Mathematical Engineering

Acknowledgements

Mes premiers remerciements vont à mes promoteurs, Pr. Pierre-Antoine Absil et Dr. Estelle Massart pour le temps qu'ils m'ont consacré, notamment durant nos nombreuses réunions, ainsi que pour leur passion inspirante et motivante du monde Riemannien.

J'aimerais également remercier Pr. John Lee et Dr. Noémie Jacquier d'avoir accepté de faire partie du jury de ce travail, et pour la rapidité de leurs réponses.

Je me dois également de remercier tous mes anciens (et actuels) enseignants – de l'école primaire à l'université – qui m'ont donné le goût des mathématiques, de la physique et de la découverte.

Finalement, je n'oublie pas mon entourage, que ce soient mes parents, mes cokoteurs, la famille Tancré chez qui une grande part de ce mémoire s'est écrit, et en particulier Florence. Merci pour votre soutien quotidien.

Contents

Chapter 1

Introduction

People suffering from a total quadriplegia have lost the use of their four limbs. One of their only way to get some autonomy of movement is by using an electrical wheelchair. In some cases, this wheelchair can be controlled with the chin, but this is not very practical. What if patients could simply *think* to the movement they want to do and the wheelchair obeyed? One possible way to accomplish this is by collecting information coming from the brain thanks to *ElectroEncephaloGraphy* (EEG). This information is then *classified* by some algorithm: does the person desire to move forwards, to turn on the right or on the left? A device that aims to accomplish this goal is called a Brain-Computer Interface (BCI) (Yger, Berar, and Lotte, 2017).

This Master's thesis is part of this general framework: we will try to classify EEG signals. Using EEG signals presents several advantages: it is cheap compared to other sensors, it is non-invasive and it has a high temporal resolution (Lotte et al., 2018). However, its major drawback is that it presents a low signal-to-noise ratio (Congedo, 2013). To address this problem, some authors have proposed to use the *covariance* matrices of the EEG signals, instead of the raw signal (Yger, Berar, and Lotte, 2017). These covariance matrices live in a set that has a particular structure that worth being taken into account. This can be done through the *Riemannian* framework. Finally, it has also been shown in the literature that *reducing the dimension* of the covariance matrices could help the classification, both in time and in performance (Yger, Berar, and Lotte, 2017).

1.1 The problem

The goal of this thesis is thus to classify EEG signals, *via* their covariance matrices, with a dimensionality reduction, in a Riemannian framework.

Classification Let us first introduce the general concept of classification on a set $\mathcal{M}$ with a set of possible *classes* $C := \{1, ..., N_C\}$. The set $\mathcal{M}$ will, in this thesis, be the set of

covariance matrices of size $D \times D$, $\mathcal{S}_{++}^{D}$. Note that the possible classes are chosen here for convenience as being the N_C first integers, but one could imagine a more exotic class set.

Given a labelled training set $TR = \{(x_i, y_i)\}_{i=1}^{n_r} \subset \mathcal{M} \times C$, the goal of the classification is to find, using the labelled training set, a function, called a *classifier*, that maps (*classify*) any point $\tilde{x} \in \mathcal{M}$ onto a label $\tilde{y} \in C$. The set of points $Tr = \{x_i\}_{i=1}^{n_r}$ will be called the *training set*. The goal is to obtain the 'best' classifier. To measure the performance of a classifier, we will use a labelled test set $\{(\tilde{x}_i, \tilde{y}_i)\}_{i=1}^{n_e} \subset \mathcal{M} \times C$. The set $Te = \{\tilde{x}_i\}_{i=1}^{n_e}$ is called the test set. Each point of this set will be classified using the estimated classifier. Then the obtained estimated labels will be compared with the true labels $\tilde{y}_i$, thanks to a particular score function.

Another requirement pertaining to the performance of the classifier that one might want to take into account is the time required by the classification method to train (*i.e.*, finding that classifier) and to classify.

Dimensionality reduction In the particular case of the classification of covariance matrices, it has been shown in the literature (Yger, Berar, and Lotte, 2017) that applying a dimensionality reduction could help both the classification performance and the time cost. Finding this dimensionality reduction will be part of the training process. More precisely, considering $\mathcal{M} \subset \mathbb{R}^{D \times D}$, we try to find a *reduction matrix* $W \in \mathbb{R}^{D \times d}$ of full rank and with $d < D$ that will map any covariance matrix $X \in \mathcal{M}$ onto

$$X_{red} = W^{\top} X W. \tag{1.1}$$

It can be shown that X_{red} is still a covariance matrix, and that with such a mapping we can reach any covariance matrix of size $d \times d$. This set can be called the *reduced set* $\mathcal{M}_{red} = \mathcal{S}_{++}^{d}$. Note that it is independent of W.

The reduction matrix will be found as the solution of an optimisation problem

$$W := \arg \min_{W' \in G} f(W'; TR), \tag{1.2}$$

where G is to be specified, f is some cost function that translates the quality of the reduction, and TR, the labelled training set. We assume in this introduction that this problem admits a solution. Usually, the cost function f will depend on W' and TR via some distance function

$$\delta : \mathcal{S}_{++}^{d} \times \mathcal{S}_{++}^{d} \to \mathbb{R}. \tag{1.3}$$

In other words, the cost f is a function of $\delta(W'^{\top} X_i W', W'^{\top} X_j W')$ for all $(X_i, X_j) \in Tr \times Tr$.

Riemannian framework The set of points that we try to classify, the set of covariance matrices $\mathcal{S}_{++}^d$, has a particular structure that we will try to respect, using the concept of Riemannian manifold. This concept will mainly bring a suitable notion of distance, that will be used as δ (equation (1.3)), and optimisation techniques to solve problem (1.2).

Transfer learning A final important concept worth introducing is the *transfer learning*. The training data usually comes from a small training session that the patient needs to perform each time he wants to use the BCI (this is mainly due to the fact the EEG has a high cross-session variability). Reducing this training time could therefore be of a great utility. To do this, we use data coming from another session or from another subject to complete the dataset. This is what we call transfer learning. This will be done *via* a method called Riemannian Procrustes Analysis, that will be introduced in chapter 3.

Summary Figure 1.1 presents the general procedure. We have labelled training dataset, that we transform into covariance matrices. It allows to learn a reduction matrix W. With that we reduce the training set and the covariance matrix of the EEG that needs to be classified. We then give these reduced matrices and the training labels to the classifier, that finally outputs an estimation of the label.

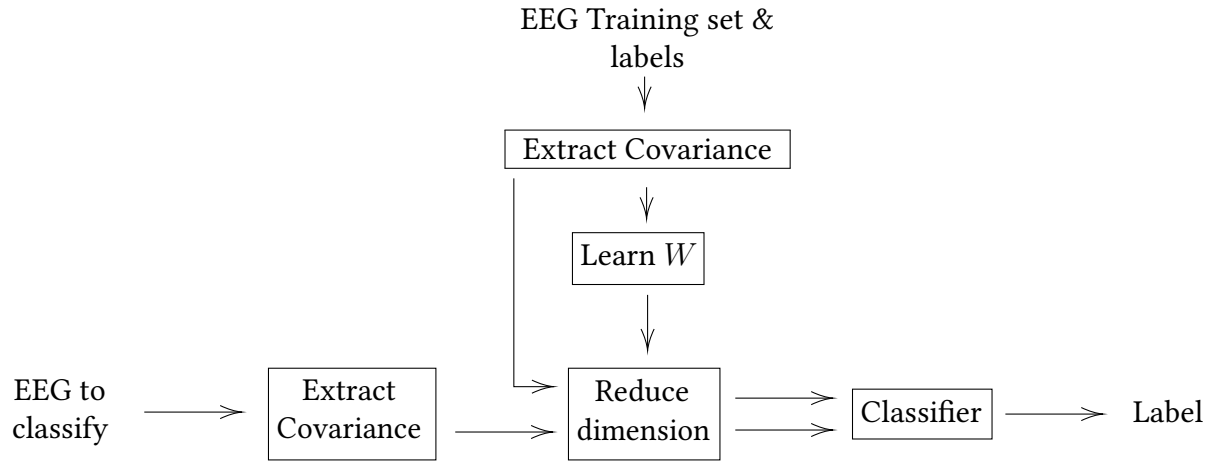

Figure 1.1: General pipeline.

Implementation The implementations in Python of the algorithms presented in this thesis mainly use the work of Vaes (2020), the Pymanopt package (Townsend, Koep, and Weichwald, 2016), the pyRiemann package (Barachant and Barthélemy, 2020) and other more usual scientific packages.

1.2 Plan of the thesis

The following chapter dives into the theoretical background that underlies this thesis: the electroencephalography, the notion of (Riemannian) manifolds and the used classifiers. It aims to clarify the general context introduced in the previous section.

Chapter 3 presents some existing classification techniques: Vaes' method, the Discriminative covariance reduction and the Riemannian Procrustes analysis, that are used further in this thesis. It ends with a small contribution, the extension of the Discriminative covariance reduction method to a multiclass problem.

The fourth chapter studies a bit further the method of Vaes and finally, chapter 5 proposes a new objective function to compute the reduction matrix W and a stochastic gradient descent approach to optimise upon this new objective function. It also compares the performance (classification and time) of these new methods with Vaes' method.

Chapter 2

Theoretical background

This chapter introduces the particular theoretical concepts used in this Master's thesis. The first section introduces the biological signals that we aim to classify and the dataset used in our experiments. The second section introduces the general concept of manifolds, the SPD manifold and the Grassmannian. Finally, section 2.3 explains the two classifiers used here (MDM and KNN) and the way we will evaluate their performances. The reader who feels comfortable with all these notions may jump to chapter 3.

2.1 Electroencephalograms (EEG)

The goal of a brain computer interface (BCI) is to allow a person to control a device, through a computer, without having to make any physical movement (Vallabhaneni, Wang, and He, 2005). An evident example of application is a brain-controlled wheelchair. To be able to do this, a BCI should have a source of information. In this thesis, the source of information will be *electroencephalograms* (EEGs). An EEG is a record of the brain electrical activity that is measured trough electrodes that are posed directly on the scalp of a person (Nunez and Srinivasan, 2007). This system has some considerable advantages (Rodrigues, Jutten, and Congedo, 2019):

- it is cheap compared to other neuro-physiological sensors,
- it is non-invasive,
- it has a high temporal resolution.

This is however counterbalanced by some negative aspects:

- it presents a low signal-to-noise ratio
- and a low spatial resolution (Congedo, 2013),
- it has a high cross-session and cross-subject variability (Lotte et al., 2018).

In this thesis, we will focus on EEGs that capture *steady state visually evoked potentials* (SSVEPs). SSVEPs are natural neural responses that are observed in the occipital area of the brain when the retina is subject to a visual stimulation pulsing at a particular frequency situated between $3.5\ Hz$ and $75\ Hz$. The frequency of an SSVEP and of its visual stimulation will be equal (or modulo an integer) (Beverina et al., 2003).

The next section describes the particular dataset that has been used for our numerical experiments.

2.1.1 Description of the dataset

The dataset used in this thesis is introduced in Kalunga et al. (2014) and available on the following GitHub repository Chevallier (2020b). It contains the recordings for 12 subjects. EEGs have been recorded at $256\ Hz$ on 8 channels.

The visual stimuli used to trigger SSVEPs were flashing at $13\ Hz$, $17\ Hz$ and $21\ Hz$. Recordings have also been made without any visual stimulus. The 3 frequencies and the rest state correspond thus to the four classes that we will use in the classification process.

The eight electrodes corresponding to the eight channels have been placed on the occipital region (on Oz, O1, O2, POz, PO3, PO4, PO7 and PO8 according to the 10/20 system, see figure 2.1).

From these recordings, for each subject and each class, 16 epochs of 3 seconds have been extracted spaced by a 0.5 second break, but for subject 10 (32 epochs) and subject 12 (40 epochs). All these epochs have then been filtered between $12\ Hz$ and $45\ Hz$ to remove the noise, while allowing the SSVEP frequencies. Finally, each epoch is filtered around the three SSVEP frequencies, such as done in Chevallier (2020a).

Each epoch will then give birth to a matrix $Y \in \mathbb{R}^{3n \times m}$ where m is the sampling size ($3\ s \times 256\ Hz$) and $n = 8$ is the number of electrodes. The latter is multiplied by three because of the three filters around the stimulus frequencies.

The matrices Y are then transformed into covariance matrices. A simple way to this is to consider the sample covariance matrix

$$\Sigma = \frac{1}{m-1} Y Y^\top. \tag{2.1}$$

In practice, the latter are estimated with a more complex method, the shrunk Ledoit-Wolf covariance matrix (Ledoit and Wolf, 2004). One of the advantages of this method is that it produces better conditioned matrices, which is an asset in this thesis where the covariance matrices will be inverted or passed through the matrix logarithm function. This requires well conditioned matrices to avoid numerical errors or an explosion of the initial bias on the covariance matrix.

These covariance matrices, together with their labels (rest or stimulus frequency) will constitute the dataset used in our experiments. We also keep track of the source subject

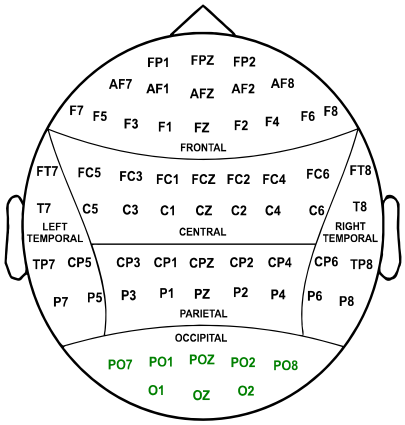

Figure 2.1: Extended 10-20 system for EEG recording. In green the electrodes used to build the dataset of Chevallier (2020b). Image based on Kamarajan et al. (2015).

for each covariance matrix, because, as explained before, EEGs have a high cross-subject variability (we can thus consider that we actually have 12 datasets).

2.2 Manifolds

With the concept of *Euclidean spaces* come several powerful tools that can be used for diverse purposes, such as the gradient, the distance, the inner product and so on. Alas, this concept of Euclidean space requires by definition that this set is closed under linear combination of its elements (vector space). A lot of interesting sets do not fulfil this requirement. For instance, the unit sphere $S^2 := \{x \in \mathbb{R}^3 \mid \|x\|_2 = 1\}$ or the set of 2×2 symmetric positive definite (SPD) matrices $\mathcal{S}^2_{++}$ are not vector spaces. Indeed, $x = (0, 0, 1) \in S^2$ and $y = (0, 0, -1) \in S^2$ but $x + y = (0, 0, 0) \notin S^2$ and

$$M = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \in \mathcal{S}^2_{++} \qquad \text{but} \qquad -1 \times M = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \notin \mathcal{S}^2_{++}.$$

Both of these sets are used in many applications. It may thus be quite useful to try to leverage the concepts born from the Euclidean spaces to these nonlinear spaces. This is precisely what the theory of *manifolds* does. This section aims to introduce basic notions used in this thesis. For more information the reader may refer to the the following books, which are the main sources for this section: Tu (2010), Lee (2018) and Boumal (2020). The latter is particularly adapted for engineers because it has a more practical approach. However, in this section, we will build the concepts one by one. We will firstly follow Tu (2010, section 5). A first concept to introduce is the smooth manifold. Intuitively, smooth manifolds are the generalisation of smooth curves and smooth surfaces to a possibly higher dimension.

2.2.1 Smooth manifolds

We will directly start with the notion of topological manifold.

Definition 2.1 (*Topological manifold*)

A *topological manifold* of dimension n is a topological set that is

- *locally Euclidean*, meaning that at every point, a neighbourhood exists that is homeomorphic to an open subset of $\mathbb{R}^n$ – this homeomorphism is called a *chart* (this condition is the one that defines a manifold),
- *Hausdorff*, *i.e.*, for each pair of distinct points, there exist two neighbourhoods (one for each point) that do not intersect,
- *second-countable*, *i.e.*, its topological space admits a countable base for its open sets.

Note that this definition allows a lot of spaces to be manifolds (for example the contour of a square) but excludes also some possibly interesting sets, such as closed balls or the set of matrices of bounded-rank (actually they are manifolds with boundaries) or such as the lemniscate of Bernoulli (see figure 2.2: everywhere but at the origin their is a neighbourhood homeomorph to $\mathbb{R}$, but at the origin, the intersection makes an homeomorphism with $\mathbb{R}$ impossible).

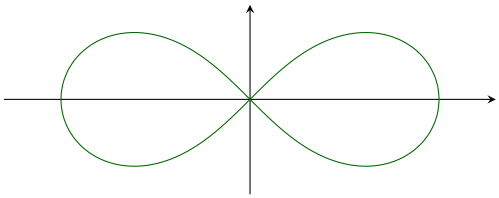

Figure 2.2: The lemniscate is not a manifold.

Let us complete the small note about the charts given in definition 2.1. Consider a locally Euclidean set $\mathcal{M}$, with at point $x \in \mathcal{M}$, a neighbourhood U homeomorph to an open subset of $\mathbb{R}^n$ trough the homeomorphism $\phi : U \to \mathbb{R}^n$. We call the pair (U, ϕ) a chart, with U the coordinate neighbourhood and ϕ the coordinate map. Two different charts may have intersecting coordinate neighbourhoods. If, at this intersection, the coordinate maps behave nicely (in a way that will be defined), the manifold is smooth.

Definition 2.2 ($\mathcal{C}^\infty$*-compatible charts*)

Two charts $(U, \phi : U \to \mathbb{R}^n)$ and $(V, \psi : V \to \mathbb{R}^n)$ of a topological manifold are said to be $\mathcal{C}^\infty$*-compatible* if the two maps

$$\phi \circ \psi^{-1} : \psi(U \cap V) \to \phi(U \cap V) \quad \text{and} \quad \psi \circ \phi^{-1} : \phi(U \cap V) \to \psi(U \cap V)$$

are of class $\mathcal{C}^\infty$. These two maps are called the *transition functions* between the charts. If U and V do not intersect, the charts are defined to be $\mathcal{C}^\infty$-compatible.

Note that the inputs and outputs of the transition functions are subsets of $\mathbb{R}^n$. Other definitions of compatibility may be built, using other classes than $\mathcal{C}^\infty$, but as the $\mathcal{C}^\infty$ class is the only one considered here, we will speak of compatible chart without the '$\mathcal{C}^\infty$', and we will do the same for the $\mathcal{C}^\infty$-atlas.

Definition 2.3 (*Atlas*)

An *atlas* on a topological manifold $\mathcal{M}$ is a collection $\mathfrak{U} = \{(U_i, \phi_i)\}$ of charts that are pairwise compatible and whose coordinate neighbourhoods cover the whole manifold, *i.e.*, $\cup_i U_i = \mathcal{M}$.

An atlas $\mathfrak{A}$ is said to be *maximal* if for any atlas $\mathfrak{B}$ such that $\mathfrak{A} \cup \mathfrak{B}$ is an atlas, [1] $\mathfrak{B} \subseteq \mathfrak{A}$. With that in hand, we can define the smooth manifold.

Definition 2.4 (*Smooth manifold*)

A *smooth manifold* is a topological manifold endowed with a maximal atlas.

Actually, any atlas on a topological manifold is contained in a unique maximal atlas. To check whether a topological space is a smooth manifold it is then sufficient to check that it is Hausdorff and second countable, and that it admits an atlas.

2.2.2 Riemannian manifolds

The notion of distance between two points is quite important in general and more particularly in this thesis. But what is exactly a distance? This concept is well-defined in mathematics (Boumal, 2020).

Definition 2.5 (*Distance*)

A *distance* on a set $\mathcal{M}$ is a function $d : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ such that for any $x, y, z \in \mathcal{M}$,

1. $d(x, y) = d(y, x)$ (it is symmetric),
2. $d(x, y) \geq 0$ (it is positive semi definite) and $d(x, y) = 0 \Leftrightarrow x = y$ (it is separable),
3. $d(x, z) \leq d(x, y) + d(y, z)$ (triangular inequality).

From a more informal and intuitive perspective, another 'definition' [2] could be as follows: *the distance between two points is the length of the shortest curve between them*, where 'shortest curve' means that this curve has the smallest length of all the possible 'reasonable'

[1] Which may not be the case, as the charts need to be compatible.

[2] This 'definition' lacks rigour and does not pretend to be the exact definition of the notion of distance.

curves. To compute this distance, we thus need to be able to compute the length of a curve. In a Euclidean space $\mathcal{E}$, the length of a piecewise smooth curve segment (see Boumal, 2020 for a precise definition) $c : [a, b] \to \mathcal{E}$ can simply be computed by integrating the *speed* of the curve

$$L_e(c) := \int_a^b \|c'(t)\| \mathrm{d}t, \tag{2.2}$$

where the norm $\|\cdot\|$ is here the norm of the Euclidean space $\mathcal{E}$. The derivative of the curve c lives here in the same space as c, but in the manifold theory, it is not always the case: let us consider for instance $\mathcal{M} = \{x \in \mathbb{R}^2 \mid \|x\| = 1\}$ the unit circle, and the curve

$$c : [0; \pi] \to \mathcal{M} \tag{2.3}$$

$$t \mapsto c(t) = (\cos(2t), \sin(2t)). \tag{2.4}$$

The derivative of c (see Boumal, 2020 for more details) is given by

$$c'(t) = (-2\sin(2t), 2\cos(2t)),$$

whose norm is greater than 1 and so does not live in $\mathcal{M}$.

However, we can see that this derivative seems to live in a space that is tangent to the manifold at the considered point, see figure 2.3. The tangent space to a manifold $\mathcal{M}$ at point $x \in \mathcal{M}$ will be denoted $T_x\mathcal{M}$ and is intuitively the vector space of the velocities at point x of all the smooth curves passing through x. Another way to see this tangent space is to consider it as a linear approximation of the manifold $\mathcal{M}$ at point x. Again a more precise definition can be found in Boumal (2020). We can also define the *tangent bundle* as being $T\mathcal{M} := \{(x, v) : x \in \mathcal{M}, v \in T_x\mathcal{M}\}$.

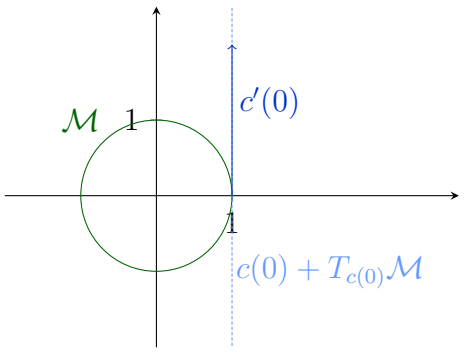

Figure 2.3: An example of tangent space.

It remains to adapt the definition of the norm used in equation 2.2. As we have seen, the velocity lives in the tangent (vector) space. We can thus define this norm *via* an inner product on the tangent space (a bilinear, symmetric, positive definite function). The

definition of this inner product may vary with $x \in \mathcal{M}$, since the tangent space may vary with x. This variation must however happen 'smoothly', as defined hereafter. The inner product between tangent vectors at x will be denoted by the following application

$$\langle \cdot, \cdot \rangle_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}. \tag{2.5}$$

Its induced norm for a tangent vector $u \in T_x\mathcal{M}$ will be $\|u\|_x := \sqrt{\langle u, u \rangle_x}$. A collection of inner products $\langle \cdot, \cdot \rangle_x$ for all $x \in \mathcal{M}$ is called a *metric*. Note that in some references, the term *metric* refers to a *distance*. This association will also be made later in this thesis.

Definition 2.6 (*Riemannian metric*)

A metric on a smooth manifold $\mathcal{M}$ composed by the inner products $\langle \cdot, \cdot \rangle_x$ for $x \in \mathcal{M}$ is said to be *Riemannian* if it varies smoothly with x, *i.e.*, if $V : \mathcal{M} \to T_x\mathcal{M}$, $W : \mathcal{M} \to T_x\mathcal{M}$ are two smooth vector fields then the function $x \mapsto \langle V(x), W(x) \rangle_x$ is smooth from $\mathcal{M}$ to $\mathbb{R}$.

Definition 2.7 (*Riemannian manifold*)

A smooth manifold endowed with a Riemannian metric is called a *Riemannian manifold*.

We can now rewrite (2.2) in a Riemannian form, given a piecewise smooth curve segment $c : [a, b] \to \mathcal{M}$

$$L(c) := \int_a^b \|c'(t)\|_{c(t)} \mathrm{d}t. \tag{2.6}$$

We now have everything in hands to define the Riemannian distance.

Definition 2.8 (*Riemannian distance*)

Given a Riemmanian manifold $\mathcal{M}$, its given metric $\langle \cdot, \cdot \rangle_x$, the induced *Riemannian distance* between $x, y \in \mathcal{M}$ is

$$d(x, y) = \inf_c L(c), \tag{2.7}$$

where the infimum is taken over all piecewise regular curve segments on $\mathcal{M}$ connecting x to y (see Boumal, 2020 for a precise definition).

Note that if the manifold is not connected, the set on which the infimum is taken may not exist and in that case the distance may be defined as being infinite, but in this thesis we always consider connected manifolds.

If the infimum in definition 2.8 is reached by some curve c, it corresponds to a *geodesic* (up to a constant speed parametrisation, [3] see Lee (2018)) which is often denoted by γ. In

[3] Such that the length of any compact subinterval I of its domain is proportional to the length of the arc that I defines.

the case of an embedded manifold of $\mathbb{R}^n$, geodesics are the (smooth) curves that present an extrinsic acceleration [4] everywhere normal to the manifold. For a sphere, these curves correspond to the great circles (equator and meridians) parametrised with a constant speed. Clearly, their second derivatives computed in $\mathbb{R}^3$ are normal to the sphere.

From another point of view and considering the surfaces (the same thought experiment could be made in other dimensions), they also correspond to the trajectories that a ball could take when rolling on the surface, if we do not consider any other forces than the one that keeps the ball on the surface, which can only act normally to the surface, hence so does the acceleration of the ball.

A particular type of geodesics is interesting: one can show (Lee, 2018) that for every $(x, v) \in T\mathcal{M}$, there exists a unique *maximal* geodesic

$$\gamma_v : I \in \mathcal{M} \tag{2.8}$$

$$\text{such that} \quad \gamma_v(0) = x \tag{2.9}$$

$$\gamma_v'(0) = v, \tag{2.10}$$

with *maximal* meaning that I is the largest as possible (such as the maximal in 'maximal integral curves'). If $[0, 1] \subseteq I$, one can then define:

Definition 2.9 (*Exponential map*)

The *exponential map* is the function

$$\{(x, v) \in T\mathcal{M} \mid [0, 1] \subseteq \operatorname{dom} \gamma_v\} \to \mathcal{M} \tag{2.11}$$

$$(x, v) \mapsto \operatorname{Exp}(x, v) = \operatorname{Exp}_x(v) = \gamma_v(1). \tag{2.12}$$

Finally, an important concept to construct an optimisation algorithm on a Riemannian manifold is the *retraction*, that allows to 'retract' a tangent vector on the manifold.

Definition 2.10 (*Retraction*)

A *retraction* on $\mathcal{M}$ is a smooth map $\mathrm{R} : T\mathcal{M} \to \mathcal{M}$ such that for any $x \in \mathcal{M}$, its restriction at x, $\mathrm{R}_x : T_x\mathcal{M} \to \mathcal{M}$ has the following properties:

1. $\mathrm{R}_x(0) = x$
2. $\mathrm{D}\,\mathrm{R}_x[v] = v$, for all $v \in T_x\mathcal{M}$

where $\mathrm{D}\,\mathrm{R}_x[v]$ denotes the differential of R_x evaluated at v (see Boumal, 2020).

Note that the exponential map is a retraction. Retractions are particularly useful when we are at a particular point of the manifold and we want to move in a certain direction

[4] For an embedded manifold in $\mathbb{R}^n$, it corresponds to the second derivative when seeing the curve in $\mathbb{R}^n$.

while staying on the manifold. Such a procedure is exactly what is done in a gradient descent (see for example algorithm 2).

2.2.3 The Symmetric Positive Definite (SPD) manifold

As explained before, we will classify the EEG signals using their corresponding covariance matrices (considering a centred EEG signal $X \in \mathbb{R}^{D\times m}$ where D is the number of channels in the signal and m is the size of the sample, the sample covariance matrix of this signal is $XX^\top/(m-1)$). Covariance matrices can easily be shown to be symmetric. Another interesting property of the covariances matrices is that they are positive definite, when X has full rank. Indeed, considering some vector $x \in \mathbb{R}^D$ with $x \neq 0$,

$$x^\top \left(\frac{1}{m-1}XX^\top\right) x = \frac{1}{m-1}(x^\top X)(X^\top x) \tag{2.13}$$

$$= \frac{1}{m-1}(X^\top x)^\top (X^\top x) \tag{2.14}$$

$$= \frac{1}{m-1}\|X^\top x\|^2 \tag{2.15}$$

and $\|X^\top x\|^2 > 0$. The fact that $\|X^\top x\|^2 \neq 0$ comes from the fact that X is assumed to have full rank. The covariance matrices are thus in the symmetric positive definite set. It can also be shown by the Cholesky factorisation that the SPD set is in the set of covariance matrices, *i.e.*, for an SPD matrix $C \in \mathbb{R}^{D\times D}$, there exists a matrix $X \in \mathbb{R}^{D\times m}$, such that $C = \frac{1}{m-1}XX^\top$. Both sets are thus equal.

It turns out that the SPD set is actually a smooth manifold (see Boumal, 2020), what is particularly convenient. The smooth manifold of symmetric positive definite matrices of size $D \times D$ will be noted $\mathcal{S}^D_{++}$. It remains to provide it with a Riemannian metric, or with a notion of distance. We will present four possible distances.

1. Euclidean distance

A quite direct notion of distance for matrices is the Euclidean distance, defined thanks to the Frobenius inner product between $A, B \in \mathbb{R}^{n\times n}$

$$\langle A, B\rangle_{\mathcal{F}} := \mathrm{tr}(A^\top B), \tag{2.16}$$

and the Frobenius norm:

$$\|A\|_{\mathcal{F}} := \sqrt{\langle A, A\rangle_{\mathcal{F}}}, \tag{2.17}$$

which is simply the square root of the sum of the square of the elements of A.

This notion of norm induces a distance.

Definition 2.11 (*Euclidean distance*)

The *Euclidean distance* between two matrices $X_1, X_2 \in \mathcal{S}_{++}^n$ is given by

$$\delta_e(X_1, X_2) := \|X_1 - X_2\|_{\mathcal{F}}. \tag{2.18}$$

This distance has shown to deliver worse results in the classification process implemented by Vaes (2020) than the other metrics. However we still present it as it is a well-known distance. The cause of those poor performances are maybe due to the fact that the Euclidean metric takes less into account the particular geometry of the SPD manifold (see the example in the second presented metric).

2. Affine Invariant Riemannian Metric

Another interesting metric is the Affine Invariant Riemannian Metric (AIRM) that can be defined as follows (Boumal, 2020).

Definition 2.12 (*AIRM*)

On any point $X \in \mathcal{S}_{++}^n$ the *Affine Invariant Riemannian Metric* presents the following inner product between two tangent vectors $U, V \in T_X\mathcal{S}_{++}^n$

$$\langle U, V\rangle_X^{aff} := \langle X^{-1/2}UX^{-1/2}, X^{-1/2}VX^{-1/2}\rangle_{\mathcal{F}} = \operatorname{tr}(X^{-1}UX^{-1}V) \tag{2.19}$$

where $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ is the Frobenius inner product (see equation (2.16)). Its induced distance between two matrices $X_1, X_2 \in \mathcal{S}_{++}^n$ is given by

$$\delta_a(X_1, X_2) := \|\log(X_1^{-1/2}X_2X_1^{-1/2})\|_{\mathcal{F}} \tag{2.20}$$

with $\log$ the principal matrix logarithm and $\|\cdot\|_{\mathcal{F}}$ the Frobenius norm (see equation (2.17)).

The origins of the name of this metric come from the fact that, for $X \in \mathcal{S}_{++}^n$ and any $M \in \mathbb{R}^{n\times n}$ that is invertible (this implies that $MXM^\top \in \mathcal{S}_{++}^n$), we have

$$\langle MUM^\top, MVM^\top\rangle_{MXM^\top}^{aff} = \langle U, V\rangle_X^{aff}. \tag{2.21}$$

Note that a big difference may be observed between the Euclidean distance and the other metrics. We will illustrate this with a small example on $\mathcal{S}_{++}^2$. First, note that we need only three 'variables' to describe $\mathcal{S}_{++}^2$: the two elements on the diagonal and the one off the diagonal. This implies that we could represent $\mathcal{S}_{++}^2$ in $\mathbb{R}^3$. For that, let us consider a general matrix

$$X = \begin{pmatrix} x & y \\ y & z \end{pmatrix} \tag{2.22}$$

with $x, y, z \in \mathbb{R}$. In order to have $\mathcal{S}^2_{++}$, one has to impose that the eigenvalues of X are positive. This may be done by imposing that the determinant and the trace of X are both positive, or in other words

$$X \in \mathcal{S}^2_{++} \Leftrightarrow \begin{cases} x + z > 0 \\ xz - y^2 > 0. \end{cases}$$

These two equations define an open convex cone in $\mathbb{R}^3$. Its boundary can be easily identified and corresponds to the set defined by

$$\mathcal{C} := \{(x, y, z) \in \mathbb{R}^3 \mid xz - y^2 = 0 \text{ and } x, z \geq 0\}. \tag{2.23}$$

If we go back to the matrix world, this set is associated to the matrices with two eigenvalues $(\lambda_1, \lambda_2) \in 0 \times \mathbb{R}_{++}$ (thus with $\lambda_2 > 0$). This set is presented on figure 2.4.

Let us now consider

$$X_1 = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix} \qquad X_2 = \begin{pmatrix} 1 & -0.9 \\ -0.9 & 1 \end{pmatrix}. \tag{2.24}$$

They both are SPD. Figure 2.4 shows the representation of the (segment of a) geodesic defined by the AIRM and that passes trough X_1 and X_2 in red. [5] It also represents in green the Euclidean 'shortest path' between these two points, *i.e.*, the line segment $\lambda X_1 + (1 - \lambda)X_2$ for $\lambda \in [0, 1]$. One can see graphically that there is a big difference between these two curves, but one should keep in mind that this is a representation of the $\mathcal{S}^2_{++}$ manifold, so that relevant difference could be due to the representation itself. In order to quantify this difference, let us compute the mean [6] between X_1 and X_2.

For the AIRM, we obtain

$$\mu_{aff} = \begin{pmatrix} 0.4359 & 0 \\ 0 & 0.4359 \end{pmatrix}$$

while for the Euclidean distance, we obtain

$$\mu_{euc} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.25}$$

The difference between these values is not negligible at all. One particular point of comparison is the determinant of these means: $\det \mu_{aff} = \det X_1 = \det X_2 = 0.19$

[5] An analytical expression of the AIRM geodesic that links two points can be found in Boumal (2020).

[6] The notion of mean, or barycenter is precisely defined in the next section. For now, we can consider that the mean is at the point at half-way on the geodesic/path that links the two points.

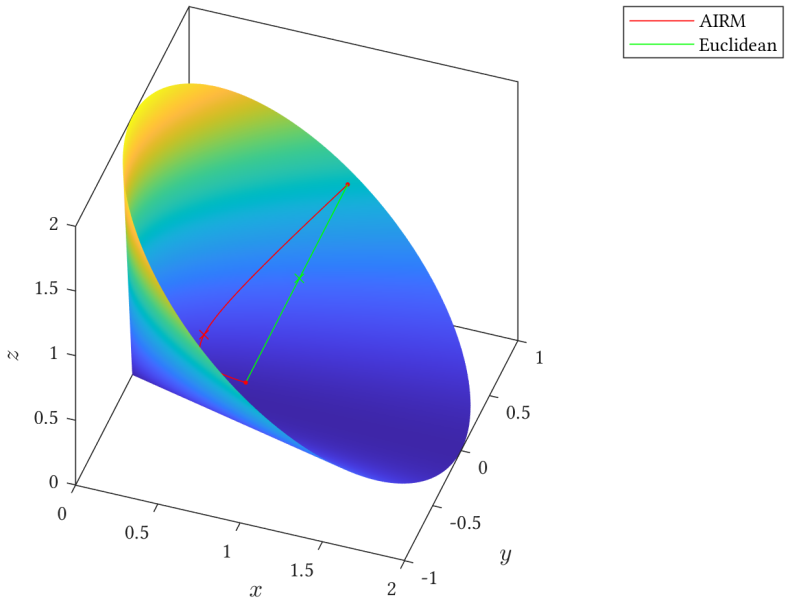

Figure 2.4: The cone represents the boundary of $\mathcal{S}^2_{++}$ mapped on $\mathbb{R}^3$, as defined in equation (2.23). In red, the AIRM geodesic between X_1 and X_2 defined in equation (2.24) with its mean (the red $\times$) and in green the Euclidean equivalent.

while $\det \mu_{euc} = 1$. In the case of the (sample) covariance matrix of a signal $X \in \mathbb{R}^{D\times d}$, the determinant is linked to the dispersion in $\mathbb{R}^d$ of the lines of X (if X is an EEG signal, a line corresponds to the data of an electrode). In this thesis, we use the mean to represent a class (in the MDM classifier, introduced in the next section). If this mean has a significantly larger determinant than the points that it tries to represent, it is maybe not a very good representative.

3. Log-Euclidean metric

A third metric that can be considered is the Log-Euclidean metric, studied in Arsigny et al. (2007). It can be obtained from the Euclidean metric that has been pulled back from the set of symmetric matrices to the set of SPD matrices.

Definition 2.13 (*Log-Euclidean metric*)

On any point $X \in \mathcal{S}^n_{++}$, the *Log-Euclidean metric* presents the following inner product between two tangent vectors $U, V \in T_X\mathcal{S}^n_{++}$

$$\langle U, V\rangle^{\log}_X := \langle \mathrm{D}\log(X)[U], \mathrm{D}\log(X)[V]\rangle, \tag{2.26}$$

where $\mathrm{D}\log(X)$ is the differential of the principal matrix logarithm at point X. Its

induced distance between two matrices $X_1, X_2 \in \mathcal{S}_{++}^n$ is given by

$$\delta_l(X_1, X_2) := \| \log(X_1) - \log(X_2) \|_{\mathcal{F}} \tag{2.27}$$

with $\log$ the principal matrix logarithm and $\| \cdot \|_{\mathcal{F}}$ the Frobenius norm.

4. Log-Det metric

Another important function that has been proven to be a distance on $\mathcal{S}_{++}^n$ by Sra (2012) is the so-called Log-Det metric. It actually corresponds to the square root of the Stein divergence (Nielsen, 2016).

Definition 2.14 (*Log-Det metric*)

The *Log-Det metric* between two matrices $X_1, X_2 \in \mathcal{S}_{++}^n$ is given by

$$\delta_{ld}(X_1, X_2) := \sqrt{\log\det\left(\frac{X_1 + X_2}{2}\right) - \frac{1}{2}\log\det(X_1 X_2)}. \tag{2.28}$$

This distance has the advantage to be computationally less expensive than the distance induced by the AIRM, while taking more into account the geometry of the SPD manifold than the Euclidean distance.

2.2.4 The Grassmann manifold

Another important manifold is the Grassmann manifold. It is used throughout this thesis as the set on which the optimisation problem to find the reduction matrix W is defined. Basically, the Grassmann manifold of height d and width [7] k is the set of all the subspaces of dimension k in $\mathbb{R}^d$, but we will use another definition, equivalent and more practical to use with matrices. In order to define it, we start from the Stiefel manifold (Boumal, 2020).

Definition 2.15 (*Stiefel manifold*)

The *Stiefel manifold* of height d and width k is the set

$$\mathrm{St}(d, k) := \{U \in \mathbb{R}^{d \times k} \mid U^\top U = I_k\},$$

where I_k is the identity matrix of size k.

The Stiefel manifold is thus the set of all matrices of size $d \times k$ with orthonormal columns, and is embedded in $\mathbb{R}^{d \times k}$. We can see that this set is a good basis to build the new definition of the Grassmannian: an element of $\mathrm{St}(d, k)$ generates, thanks to its columns, a

[7] These terms come from Townsend, Koep, and Weichwald (2016).

subspace of dimension k in $\mathbb{R}^d$, and any possible k-dimensional subspace has a basis that corresponds to an element of $\mathrm{St}(d,k)$. There is still one small 'correction' to apply to this set in order to find the appropriate definition: two matrices of the Stiefel manifold can represent the same subspace (if they are equal up to a rotation). To solve this problem, let us define the orthogonal group

$$\mathrm{O}(k) := \{Q \in \mathbb{R}^{k\times k} \mid Q^\top Q = I_k\}. \tag{2.29}$$

Note that it is the Stiefel manifold $\mathrm{St}(k,k)$. This orthogonal group allows to define an *equivalence relation* $\sim$ on the Stiefel manifold. For two matrices $U, V \in \mathrm{St}(d,k)$,

$$U \sim V \text{ if and only if there exists a rotation } Q \in \mathrm{O}(k) \text{ such that } U = VQ. \tag{2.30}$$

This equivalence relation induces the following *equivalence classes*

$$[U] := \{V \in \mathrm{St}(d,k) \mid U \sim V\} = \{UQ \mid Q \in \mathrm{O}(k)\}. \tag{2.31}$$

An equivalence class corresponds thus to the set of all the orthogonal basis of a given subspace. The Grassmann manifold is therefore the set of the equivalence classes. This is often called a quotient manifold.

Definition 2.16 (*Grassmann manifold*)

The *Grassmann manifold* of height d and width k is

$$\mathrm{Gr}(d,k) := \mathrm{St}(d,k)/\mathrm{O}(k) \tag{2.32}$$
$$:= \{[U] \mid U \in \mathrm{St}(d,k)\}. \tag{2.33}$$

This definition is not the usual one as explained before but is equivalent to it.

Finally, we provide the Grassmann manifold with a distance coming from Wong (1967).

Definition 2.17 (*Grassmannian 2-distance*)

The Grassmannian 2-distance between two subspaces $U, V \in \mathrm{Gr}(m,n)$ is the two-norm of the vector of canonical angles between the two subspaces, *i.e.*,

$$\delta(U,V) = \|\cos^{-1}\sigma\|_2, \tag{2.34}$$

where σ is the* vector of singular values of the matrix $U^\top V$ and the $\cos^{-1}$ is applied element by element.

*For uniqueness reasons, let us say that the vector has increasing elements.

The 2 in the name of the distance refers to the fact that we take the two-norm. Other

distances could be built based on the canonical angles (Bendokat, Zimmermann, and Absil, 2020), but we choose this one because it is a Riemannian distance (see definition 2.8). For more information about the principal angles, the reader is referred to Jungers et al. (2020).

The reason why the optimisation problem to find the reduction matrix W occurs on this set is twofold:

- W needs to have full rank, in order to have positive definite reduced covariance matrices;
- the metrics used here (see previous section) are invariant under rotation, and two of them (Log-Det and AIRM) are affine invariant (change of basis), meaning that what really matters is *the column space* of W (two W presenting the same column space will lead to the same objective value, and to the same classification).

Note that if the used metric is not affine invariant, this justification does not hold anymore. We should therefore optimise on the set of full rank matrices.

2.3 Classifiers

In order to introduce the two classifiers used here, let us consider a general classification problem on a Riemannian manifold $\mathcal{M}$ with an associated distance $\delta : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$, with the set of classes $C = \{1, ..., N_C\}$, the training set $Tr = \{x_i\}_{i=1}^{N} \subset \mathcal{M}$ and a point $y \in \mathcal{M}$ that is to be classified, with the function $class : Tr \to C$ that gives the class of a training point.

2.3.1 Minimal Distance to the Mean (MDM)

The Minimal Distance to the Mean classifier works as follows:

1. For each class $c \in C$, compute the *average* $\mu_c \in \mathcal{M}$ of the set $\{x \in \mathcal{M} \mid class(x) = c\}$.
2. For each class $c \in C$, compute the distance between y and the average of class c $\delta_c = \delta(\mu_c, y)$.
3. Assign y to the class that has the nearest average: $\arg\min_{c \in C} \delta_c$.

The way to compute the average is still to be defined. For an Euclidean space, we can simply compute the arithmetic average. This procedure for $\mathcal{M} = \mathbb{R}^2$ is depicted on figure 2.5, where there are two classes (the blue one and the red one), and we need to classify the green star.

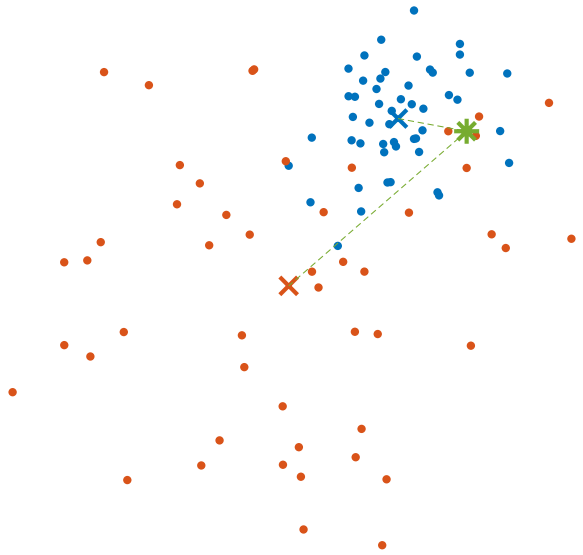

Figure 2.5: The green star will be classified as a blue dot since it is closer to the blue mean.

In a Riemannian framework, we will use the Riemannian barycenter.

Definition 2.18 (*Riemannian barycenter*)

The *Riemannian barycenter* of a set $S = \{s_i\}_{i=1}^n \subset \mathcal{M}$ where $\mathcal{M}$ is a smooth manifold endowed with a Riemannian metric δ is

$$\arg\min_{\mu \in \mathcal{M}} \sum_{i=1}^{n} \delta^2(s_i, \mu). \tag{2.35}$$

Note that the problem posed here in definition 2.18 might have a nonunique solution when the manifold has a positive curvature (see Afsari, 2011). That problem arises for example in the computation of the barycenter of two poles on a sphere (positive curvature): all the points on the equator (so an infinity of points, if the sphere's dimension is greater than or equal to 2) can be taken as a barycenter. This problem does not occur when computing a barycenter on the SPD manifold, as it has a non-positive curvature, see Bridson and Haefliger (2013).

Note that the averages of the training set may be computed once and for all in the case where more than one point is to be classified. Note also that the dispersion of each class in the training set plays an important role in the classification. Easy examples of abhorrent classification can be imagined for extreme cases.

2.3.2 K-Nearest Neighbours (KNN)

The other classification method that will be used is the k-Nearest Neighbours method (KNN). Considering again the same setting than previously, the method works as follows.

1. For each $i \in \{1, ..., N\}$, compute the distance between x_i and y: $\delta_i = \delta(x_i, y)$.
2. Select the k x_i leading to the k smallest δ_i.
3. Assign y to the most represented class in the set of the k nearest training points. If the most represented class is not unique, chose at random between all the most represented classes.

This procedure generalises quite easily from a Euclidean space to a Riemannian manifold. Following the example of the previous section, figure 2.6 shows the application of a KNN classifier with $k = 6$.

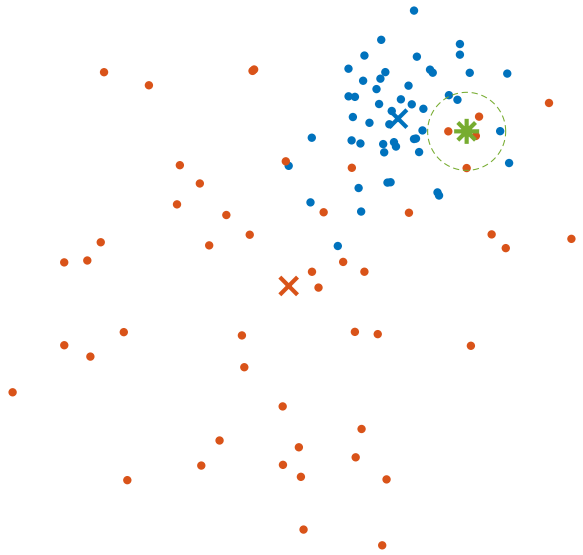

Figure 2.6: The green star will be classified as an red dot since it has 4 red neighbours and only 2 blue ones.

2.3.3 Evaluation of the classification

Once the classification is performed we obtain a vector of estimated labels y_e for the test sets and we generally want to evaluate the performance of the classifier. In this thesis, we will consider that once we have the estimated labels vector for the test set, we also have the true label vector y_t. To compare the estimated labels with the true ones, several scores exist (Mishra, 2018; Döring, 2018) but we will use the ones presented and implemented by Vaes (2020): the *F1 micro score* and the *F1 macro score*. For each class c, we can define

- TP_c the true positives, *i.e.*, the number of estimated labels that have been correctly assigned to class c,
- TN_c the true negatives, *i.e.*, the number of estimated labels that have been correctly not assigned to class c,

- FP_c the false positives, *i.e.*, the number of estimated labels that have been wrongly assigned to class c,
- FN_c the false negatives, *i.e.*, the number of estimated labels that have been wrongly not assigned to class c.

For example, consider the following vectors, assuming there are three classes $\{1, 2, 3\}$:

$$y_e = (1, 2, 2, 3, 1)$$
$$y_t = (1, 1, 3, 3, 2)$$

For class 1, the first element is a true positive, the second is a false negative, the third and fourth are a true negatives and the last one is a false positive.

We can now define the scores, considering N classes.

Definition 2.19 (*F1 micro score*)

Defining

$$P_{micro} = \frac{\sum_c TP_c}{\sum_c (TP_c + FP_c)}, \qquad R_{micro} = \frac{\sum_c TP_c}{\sum_c (TP_c + FN_c)}$$

as the micro average precision and the micro average recall, the *F1 micro score* is computed as follows

$$F1_{micro} = 2\frac{P_{micro} R_{micro}}{P_{micro} + R_{micro}}.$$

Definition 2.20 (*F1 macro score*)

Defining

$$P_{macro} = N^{-1} \sum_c \frac{TP_c}{TP_c + FP_c}, \qquad R_{macro} = N^{-1} \sum_c \frac{TP_c}{TP_c + FN_c}$$

as the macro average precision and the macro average recall, the *F1 macro score* is computed as follows

$$F1_{macro} = 2\frac{P_{macro} R_{macro}}{P_{macro} + R_{macro}}.$$

One interesting remark is that both scores are equal when each class has the same number of test points, which will always be the case in this thesis, hence we will usually use one of the two. For a deeper discussion of the scores, the reader is referred to Vaes (2020).

Chapter 3

Literature review & first result

This chapter first presents some recent developments pertaining to the study. The objective is certainly not to propose a complete picture but rather sketch three methods used further in this work. For a broader review, we refer to Lotte et al. (2018) and Yger, Berar, and Lotte (2017). Finally, this chapter contains a first result that is an extension of the third presented method.

In order to unify the notation, we will consider the following one (table 3.1).

Tr	The training set
n_r	The size of the training set
Te	The test set
n_e	The size of the test set
X	A covariance matrix
D	The original dimension of the covariance matrices (so $X \in \mathbb{R}^{D \times D}$)
W	The reduction matrix
W^*	The optimal reduction matrix
d	The new dimension (so $W \in \mathbb{R}^{D \times d}$)
δ	A chosen distance

Table 3.1: Notation for this chapter.

3.1 Existing methods

In this section, we introduce three methods that have been used in this thesis.

3.1.1 Vaes

In his Master's thesis, Vaes (2020) developed a reduction method inspired from Harandi, Salzmann, and Hartley (2014). The reduction problem can be written as follows.

$$W^* := \arg \min_{W \in Gr(D,d)} \sum_{i=1}^{n_r} \sum_{j=1}^{n_r} A_{ij} \delta^2(W^\top X_i W, W^\top X_j W) \tag{3.1}$$

where δ can be the Euclidean, AIRM, Log-Euclidean or Log-Det distance. The matrix A is defined as follows.

$$A := G_w - G_b$$

with

$$G_{w,ij} := 1 \text{ iff } X_i \in N_w(X_j) \text{ or } X_j \in N_w(X_i), 0 \text{ otherwise}$$
$$G_{b,ij} := 1 \text{ iff } X_i \in N_b(X_j) \text{ or } X_j \in N_b(X_i), 0 \text{ otherwise}$$

and

- $N_w(X_j)$ is the set of the v_w closest training points to X_j and of the *same* class,
- $N_b(X_j)$ is the set of the v_b closest training points to X_j and of a *different* class.

Hence, if two training points of the same (resp. different) class were close on the initial manifold, the problem (4.1) tries to bring them closer together (resp. away from each other). Note that A is computed *a priori*, *i.e.*, before reduction. In chapter 4, we investigate the case where A is computed on the reduced manifold (and thus depends on W).

One major conclusion of this thesis is that the reduction helps the classification performance and its computational cost. It also shows that the Euclidean distance delivers poor classification performance compared to the other 'metrics'.

3.1.2 Discriminative covariance reduction (DCR)

In this subsection, we consider a *binary* class problem. In Kalaganis et al. (2020), the authors propose to build a reduction matrix by selecting a certain (*a priori* unknown) number of electrodes, *i.e.*, to consider a (full rank) matrix W whose columns are columns of the identity matrix of the corresponding size. The separability criterion that needs to be minimised on the subsets of possible electrodes can be computed as follows.

- Build the Minimal Spanning Tree (MST) of the distance graph for the considered subset of electrodes (see further);
- count the number of edges that connect two nodes of different classes.

For the first point, the distance graph for the considered subset of electrodes is the complete graph of size n_r, where each node represents a training matrix and the edge that link $X_i \in Tr$ to $X_j \in Tr$ has the weight given by $\delta(W^\top X_i W, W^\top X_j W)$. The distance δ is here the one that comes from the AIRM. Note that the reduced matrix $W^\top X W$ can be easily obtained from X: it corresponds to X for which the rows and columns that do not correspond to the considered subset of electrodes have been removed.

More mathematically, considering a subset of electrode k, W_k its associated reduction matrix, MST_k the set of the edges of the minimal spanning tree of the associated distance graph, y is the (binary) vector of labels (only two classes), we can express the criterion to *maximise* over all possible k,

$$J(W_k) := n_r - \sum_{(i,j)\in MST_k} |y_i - y_j|. \tag{3.2}$$

Since they are $2^D - 1$ possible subsets of electrodes (the empty set is not considered of course), it is vain to try to solve that problem by a brute-force approach. The authors proposed to use the *cross-entropy method* (Rubinstein, 1999; Botev et al., 2013) initially developed for rare event simulation. The algorithm is based on De Boer et al. (2005) and the general idea is to associate a certain probability (initially 0.5) to each electrode and to generate a certain number of subsets of electrodes, thanks to the probability distribution, and compute their separability score. Then we recompute the probability distribution accordingly to the observed performance of each electrode: the probability for each electrode is the proportion of time that it appears in the subsets leading to the best scores. A very interesting point is that this separability criterion is comparable across dimensions.

Section 3.2 extends their work.

3.1.3 Riemannian Procrustes Analysis (RPA)

Rodrigues, Jutten, and Congedo (2019) present a method to reconcile two datasets, the *target* set and the *source* set coming from different sources (two different sessions or subjects). Their technique follows the semi supervised transfer paradigm: the labels of the source set are known, and at least one label of each class is known in the target set. We will call the matrices of the target set with known labels the landmarks. The goal is thus to apply some transformations on the dataset such that the landmarks 'become aligned' with the distribution of their respective classes in the source set. Let us introduce some

notation before explaining those transformations. The source $\mathcal{S}$ and target $\mathcal{T}$ datasets are

$$\mathcal{S} = \{(X_i, y_i)\}_{i=1}^{n_s}$$
$$\mathcal{T} = \{(\tilde{X}_i, \tilde{y}_i)\}_{i=1}^{n_t}$$

where the X_i, $\tilde{X}_i$ are the covariance matrices, y_i, $\tilde{y}_i$ the labels. As mentioned earlier, it is assumed that the $\tilde{y}_i$ are only partially known. We therefore define $\mathcal{T}_u$ and $\mathcal{T}_\ell$, the unlabelled and labelled (landmarks) partitions of $\mathcal{T}$.

To reconcile these sets, Rodrigues, Jutten, and Congedo (2019) propose to first recenter the sets on the identity matrix, then to equalise the dispersion of the sets and finally to apply a rotation to the sets. Let us give more details.

1. Recentering The idea is to shift the sets on the SPD manifold so that their means become the identity matrix (the identity matrix can be seen as the 'origin' of the SPD manifold). To do this, a first step is to compute M and $\tilde{M}$, the barycenters of the source and target set, respectively (see definition 2.18). Then, compute

$$X_i^{(rct)} = M^{-1/2} X_i M^{-1/2} \qquad \forall i \in \{1, ..., n_s\}$$
$$\tilde{X}_i^{(rct)} = \tilde{M}^{-1/2} \tilde{X}_i \tilde{M}^{-1/2} \qquad \forall i \in \{1, ..., n_t\}.$$

Note that the pairwise AIRM distances between matrices of the source (resp. target) sets are not affected by this operation. Note also that the barycenters M and $\tilde{M}$ are indeed transported on the identity.

2. Stretching Based on the fact that $\delta_a^2(X^s, I) = s^2 \delta_a^2(X, I)$, with δ_a the AIRM and I the identity matrix, we can stretch the target dataset:

$$\tilde{X}_i^{(str)} = \left(\tilde{X}_i^{(rct)}\right)^{\sqrt{d/\tilde{d}}},$$

with d and $\tilde{d}$ the sum of the squared distances to the mean (the identity). Both sets have now the same dispersion.

Let us now enter into the supervised part of the process.

3. Rotation Because $\delta_a(U^\top X U, U^\top U) = \delta_a(X, I)$ for any orthogonal matrix U, we can interpret the operation $X \mapsto U^\top X U$ as a rotation around the identity. To find the rotation matrix U, Rodrigues, Jutten, and Congedo (2019) propose to solve an optimisation problem that consists in minimising the sum of the squared distances between a modified version of the class means of the target set and a modified and rotated version of the corresponding class mean of the source set. For more details, we refer to the article.

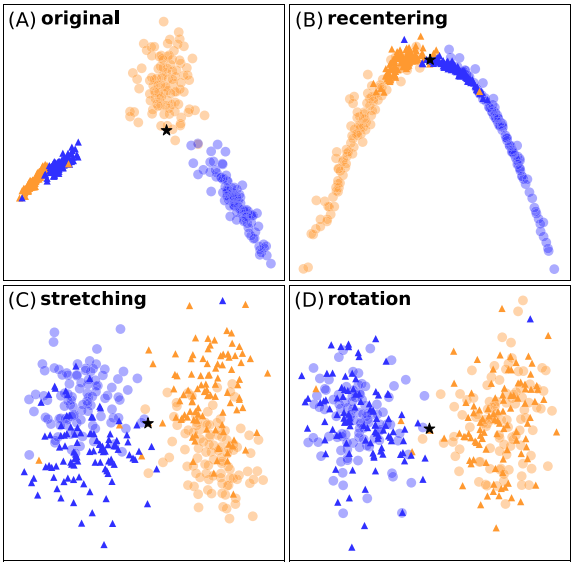

Figure 3.1: Illustration of the RPA coming from the original article (Rodrigues, Jutten, and Congedo, 2019). The filled dots with some transparency are the target dataset, the triangle are the source dataset. The two colours represent the two classes. The star is the identity matrix. The visualisation is obtained using diffusion maps (see Lafon and Lee, 2006).

Figure 3.1 shows an illustration of this procedure. Note that the RPA is based on the Procrustes Analysis, working on a Euclidean space, and used in many fields (see Gower, Dijksterhuis, et al., 2004).

3.2 Extension of the DCR

The first contribution of this thesis has been to implement the work of Kalaganis et al. (2020) (namely discriminative covariance reduction or DCR) in Python and to extend it to a multi-class problem. To do this, we extend the separability criterion presented in equation (3.2)

$$J(W_k) := n_r - \sum_{(i,j)\in MST_k} [y_i = y_j]. \tag{3.3}$$

where $[y_i = y_j] = 1$ if $y_i = y_j$ and 0 either. We therefore count the number of edges connecting two nodes of different classes. This being done, we can confront this method to the method of Vaes.

Experiment For each of the 6 first subjects, its dataset is split randomly into two sets (training and test sets). The training set is used to compute W with the method of Kalaganis et al. (2020) (DCR). Then we classify the test set with a KNN or MDM classifier.

The used metric is the AIRM, everywhere in the process. This process has been repeated thrice. In the same time, we evaluated the performance of the classifier without reduction and with the reduction of Vaes, that is performed using the same dimension than the one obtained by DCR (where the new dimension is not a parameter). The results of this experiment can be seen on figure 3.2. In the case of the MDM classifier, the method of Vaes presents the best median. In the case of the KNN classifier, the method of Vaes and the DCR present similar median, but the DCR has a smaller standard deviation. The extension of the method proposed in Kalaganis et al. (2020) seems therefore to deliver interesting results. Note that the parameters used in the algorithm for maximising the separability criterion (3.3) have been tuned by hand, but it seemed during that process that they were almost optimal. The average new dimension obtained by DCR is about 13.

Note that table A.1 presents the default parameters used in this thesis, so if a parameter is not given explicitly a value somewhere in this report, it is the one stated in that table.

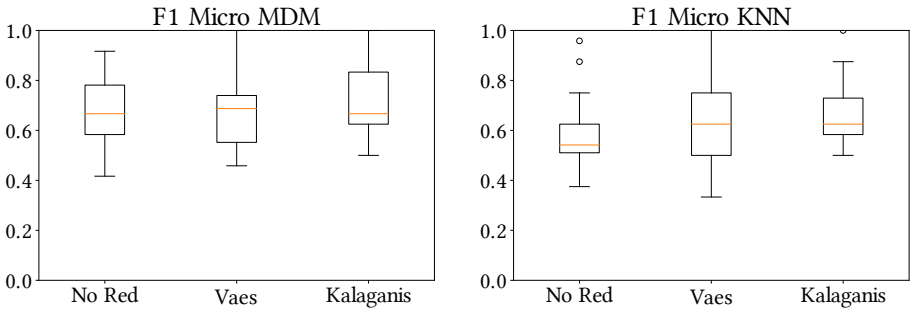

Figure 3.2: Evaluation of the method of the DCR.

Chapter 4

A further analysis of Vaes' method

This chapter aims to study the method proposed in Vaes (2020) and recalled in chapter 3. As already precised, this method tries during the optimisation process to bring closer together the nearest neighbours of the same class and to spread the nearest neighbours of different classes. The 'nearest neighbours' are computed *a priori* and thus stay the same during the whole optimisation process, but the *true* nearest neighbours may change. This could have an impact on the performances of the method. Hence, in a first time, we will study whether using the true nearest neighbours in the objective function could lead to an enhancement of the classification performances.

In a second time, inspired by Kalaganis et al. (2020) who try to apply the reduction of dimension by selecting a subset of electrodes, we will try to see whether the optimal reduction matrix W obtained by Vaes' reduction is indeed selecting some electrodes or not.

Finally, we will try to do some transfer learning with Vaes' method (*i.e.*, to use data from another session or subject as training data). A first step will be to use the raw external data as training set. Then we will try to transform the latter using a state-of-the-art method: the Riemannian Procrustes Analysis (RPA) recalled in chapter 3.

4.1 A variable affinity matrix

In Vaes (2020), the reduction matrix W is obtained by solving the following problem (see chapter 3 for more details).

$$\arg \min_{W \in Gr(D,d)} \sum_{i,j} A_{ij} \delta^2(W^\top X_i W, W^\top X_j W) \tag{4.1}$$

where D and d are respectively the dimension of the original data and the new dimension, the sum is done over the whole training set and X_i is the ith matrix of the training set. The matrix A is defined based on the situation before the reduction and this could lead to

some bad scenarios. For instance, if two points of different classes X_i and X_j were far away on the initial manifold (such that $A_{ij} = 0$), they could potentially be brought very close together on the reduced manifold. The idea of this section is thus to build A not from the initial manifold but from the reduced one. The matrix A will thus depend on W. To this end, let us define $\tilde{A}(W) := \tilde{G}_w(W) - \tilde{G}_b(W)$ and

$$\tilde{G}_{w,ij}(W) := 1 \text{ iff } X_i \in N_w(W^\top X_j W) \text{ or } X_j \in N_w(W^\top X_i W), 0 \text{ otherwise}$$
$$\tilde{G}_{b,ij}(W) := 1 \text{ iff } X_i \in N_b(W^\top X_j W) \text{ or } X_j \in N_b(W^\top X_i W), 0 \text{ otherwise}$$

Note that using $\tilde{A}$ instead of A in the minimisation problem (4.1) breaks the continuity of the objective function. In practice, the optimisation algorithm still converges, but slower.

Experiments The performances of this new method have been tested on the four first subjects with new dimension $d \in \{6, 8, 11, 15, 22\}$ three times. For each set of matrices provided by a subject, it has been divided in two, one training set and one test set, such as made in Vaes (2020) (no transfer learning thus). Table 4.1 summaries the results of this experiment. It does not allow to conclude that using $\tilde{A}$ improves significantly the classification. Figure 4.1 presents the results per dimension. It shows indeed that there is no major enhancement of the average score and that the distributions of the scores around the means are not particularly improved (except maybe for the MDM).

	A	$\tilde{A}$
MDM	0.69	0.71
KNN	0.65	0.65

Table 4.1: Average F1 micro score.

4.2 Analysis of the reduction matrix

In this section we investigate whether the reduction matrix $W \in \mathbb{R}^{D\times d}$ obtained in Vaes (2020) is 'selecting' some subset of electrodes, such as in Kalaganis et al. (2020).

Subset of electrodes Because the covariance matrices are built from three filtrations of one signal coming from 8 electrodes, choosing a subset of the 8 electrodes would correspond to taking W as a concatenation of some columns of the matrix $I_{3,8} := [I_8; I_8; I_8]/\sqrt{3}$, with ; denoting the vertical concatenation. The $\sqrt{3}$ is there to normalise the columns of $I_{3,8}$. Since what really matters is actually not W in itself but rather its column space (the optimisation happens on the Grassmannian), we can reformulate our question as: 'Is there a subset of size d of the columns of $I_{3,8}$ that generates a subspace *close to* the column space of W ?' The notion of *close to* will be evaluated by the biggest

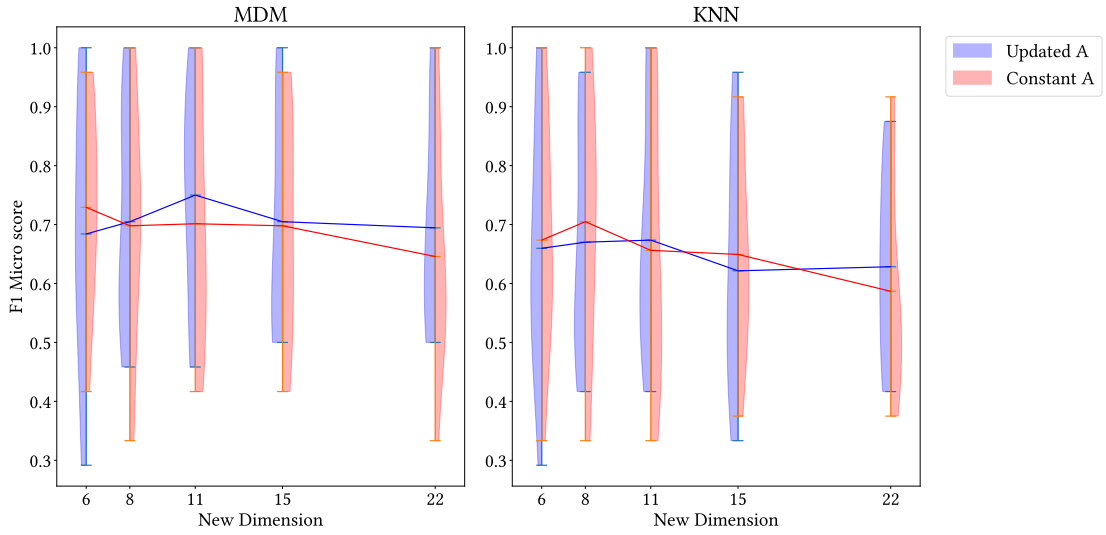

Figure 4.1: Plot of the F1 micro scores obtained on the four first subjects. Three runs have been taken for each subject for each dimension. No significant improvement is observable. The vertical areas translate the distribution of the scores.

canonical angle between the two subspaces. Denoting the matrix composed by the subset of the columns of $I_{3,8}$ by S, it can be defined as (Jungers et al., 2020)

$$\theta_{\max} := \cos^{-1}\left(\sigma_{\min}\left(W^\top S\right)\right)$$

As the matrix $I_{3,8}$ is not too large, S can be identified by a brute-force method: test all combinations of d columns and select the one with the smallest $\theta_{\max}$. We will call this minimal biggest angle θ^*. This procedure has been applied for W coming from Vaes (2020) (no transfer learning) for each of the subjects with a new dimension $d = 6$. The average θ^* is of $80.58°$ with a standard deviation of $2.97°$. In order to have a comparison point, the same procedure has been applied on a random W (chosen uniformly on the set of orthogonal matrices) 200 times and we get an average θ^* of $80.22°$ with a standard deviation of $3.64°$.

It is therefore not possible to statistically conclude that the W in Vaes (2020) selects some subset of electrodes.

Subset filtered electrodes In the previous paragraph, the same weight was attributed to each of the different frequencies of a particular electrode. In this paragraph, this assumption is not considered anymore: $I_{3,8}$ is replaced by I_{24} (this drastically increases the number of combinations to investigate in a brute-force method, but this stays manageable). The new average θ^* is now of $67.15°$ with a standard deviation of $1.53°$. For 20 random W, we obtain an average of $66.45°$ with a standard deviation of $2.06°$.

A greedy algorithm? A brute-force algorithm that tests all the possible combinations may be quite expensive. Therefore a greedy algorithm has been designed (algorithm 1):

Algorithm 1: Greedy algorithm for closest column space

For a matrix $W \in \mathbb{R}^{D\times d}$
Start from an 'empty' matrix S
for $k = 1, ..., d$ **do**
  Take C, the closest column (in the canonical angle sense) of $I_{3,8}$ (or I_{24}) from W
  Concatenate C with S
  Remove that direction from W: $W_i \leftarrow W_i - C^\top W_i C \quad \forall i \in \{1, ..., d\}$, with W_i denoting the columns of W
end
return S

Alas, that procedure does not deliver good results. The produced matrix S gives indeed a biggest canonical angle significantly greater than the one obtained by brute-force: for the greedy algorithm applied on 20 random matrices, the average θ^* is about $85.9°$ with a standard deviation of $2.9°$. Recall that the brute-force method produced an average θ^* of 66.45 in the same conditions.

Norm of the rows of W Finally, we investigated the norm of the rows of the reduction matrix W, in order to see whether the method allocates the same weight to each filtered electrode. Maybe there are always one or two prominent or discriminated rows, or maybe the same distribution of weights can be observed. The conclusion of this procedure is that no general pattern seems to stand out (see figure 4.2, subjects 15, 17 and 24 show three different plots).

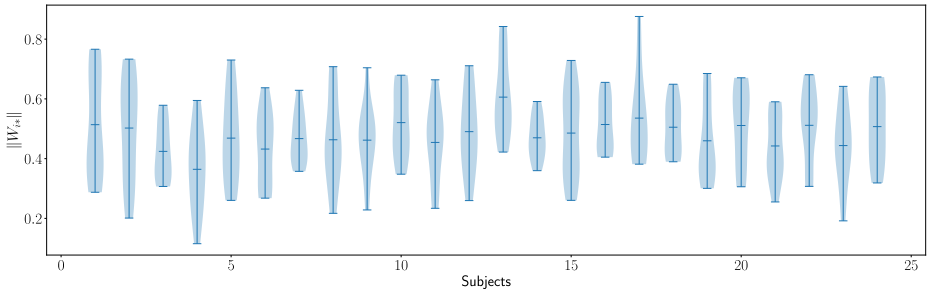

Figure 4.2: Norms of the rows of W for different subjects. The new dimension for the reduction is 6.

4.3 Vaes and transfer learning

A first part of this section will investigate the ability of Vaes' method to do transfer learning with a naive approach, then the external training set will be transformed with the Riemannian Procrutes Analysis (RPA), see chapter 3.

4.3.1 Naive approach

This section aims to evaluate the abilities of the method of Vaes (2020) for transfer learning purposes. To this end, the dataset will be divided in two. A first part of the subjects will be used as a training set and the second part as a test set.

Training set The training set is the aggregation of data coming from a given number N_{sub}^{tr} of subjects chosen randomly from the subject 1 to 8. Each subject provides a certain number of matrices for each class that will be averaged [1], in order to avoid the training set to be too large (to reduce the computational cost). Hence, *e.g.*, if we decide to take 2 subjects in the training set, we choose randomly 2 subjects between 1 to 8 (2 and 5 for example). Then for each subject and for each of the four classes, we compute the mean of the available matrices and we build the training set with it. The training set will thus be made up of 2×4 matrices. This averaged training set is used both for finding the reduction matrix W and in the classification (except in the last part of the paragraph analysing the experiments).

Test set The test set is constant: it is the union of the data coming from subjects 9 to 12.

Other parameters We used the AIRM metric, $k = \min(5, N_{sub}^{tr})$ (the main parameter of the KNN classifier, hence the number of nearest neighbours considered) and $v_b = \min(9, N_{sub}^{tr})$ (the number of nearest points of a different class considered in the building of the affinity matrix used to find the reduction matrix). The new dimension after reduction and the number of subjects in the training set have taken different values during the tests.

Experiments Figure 4.3 presents the results of the experiments. For each couple of values of new dimension and N_{sub}^{tr}, 10 tests have been run, each time with a new training set generated following the procedure established above.

This figure does not show a significant improvement of the classification with the reduction of dimension. It seems more advantageous to increase N_{sub}^{tr} instead of applying a dimension reduction (the computational cost has to be kept in mind of course).

[1] A small discussion on this choice is made at the end of this subsection.

In some cases, a jagged behaviour can be observed between odd and even new dimension. Note also that the KNN classifier seems to give worse results than the MDM classifier.

It is worth noting that the standard deviation of the samples (for each new dimension and number of subjects in the training set) divided by $\sqrt{10}$ is of the order of 1 or 2%.

In order to evaluate the impact of using an averaged training set, an experiment was carried out where no reduction was applied and the set of training matrices was given non averaged to the classifiers. The results of this test is depicted with crosses on figure 4.3. It seems that the averaging process is a little bit harmful to the MDM classifier. The gap between the dots and the crosses is expected to be small since the MDM classifier only cares about the means of the classes and the mean of the subject-means should not be far away from the global mean. Concerning the KNN classifier, the averaging process of the training matrices is beneficial to the classification.

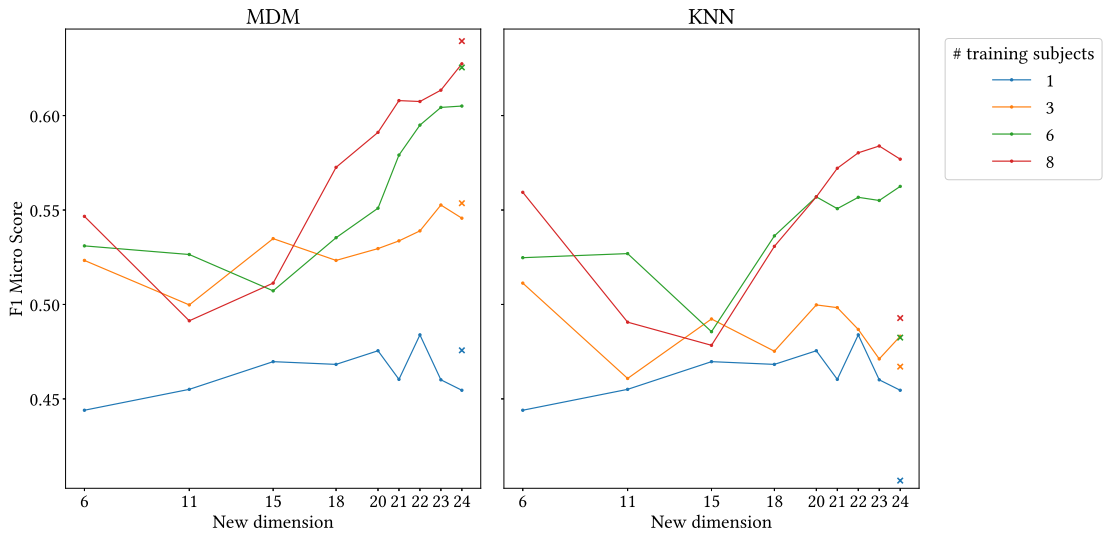

Figure 4.3: Plot of the mean F1 micro score obtained during the 10 tests described above, making the number of subjects in the training set and the new dimension vary. Concerning the case when the new dimension is 24, the matrices are kept unchanged (no transformation is performed on any matrix). The dots correspond to the score obtained when using the means of the matrices in the classifiers, while the crosses show the score obtained when the classifiers use the original set of matrices as training data.

4.3.2 Riemannian Procrustes Analysis

The idea of this section is to combine the methods of Rodrigues, Jutten, and Congedo (2019) and Vaes (2020) in order to enhance the transfer learning performances of Vaes (2020). More precisely, we will consider here a *source* subject, that will give labelled

matrices that will form the *training* set for Vaes' classification method and a *target* subject that will provide the *test* set for the classification. The target subject will also provide some labelled matrices, the *target training* set that will be only used in the RPA in order to reconcile the target and source datasets (and thus are not classified). The division of the dataset of the target subject is as follows: the 2nd, 5th and 8th matrices of each class compose the target training set and the rest compose the test set.

Experiment This procedure has been applied with subjects 1 to 4 serving individually as source subject and with subjects 7 to 10 as target subjects (thus 16 tests have been conducted). The new dimension for Vaes' method is 15 and the metric AIRM.

The means of the F1 micro scores obtained by each classification method are presented in table 4.2. These results are encouraging: the reduction of dimension seems profitable (but not significantly) and the scores are better than the ones of previous section. The scores of the classification without reduction of dimension presented in Vaes (2020) are about 0.7 for MDM and about 0.6 for KNN. Even if the conditions of the tests are not similar (here the test sets are larger, for instance), it is tempting to say that RPA does a pretty good job.

MDM	MDM with red	KNN	KNN with red
0.66	0.67	0.59	0.62

Table 4.2: Mean F1 micro score for the RPA-Vaes approach using the source set in the classification.

In this procedure, only the data from the source subject is used during the classification and not the data from the target training set. This choice is motivated by the desire to highlight that it is possible to obtain a good score with data from another subject. With the target training set included, a good score could have been due to that set. However, for a real application, it could be useful to use the target training set in the classification, with the data from the source subject. Table 4.3 presents the results of the previous experiment, but using both the source set and the target training set in the classification. We can see that the scores are indeed better.

MDM	MDM with red	KNN	KNN with red
0.71	0.73	0.65	0.70

Table 4.3: Mean F1 micro score for the RPA-Vaes approach using both the source set and the target training set in the classification.

Chapter 5

A new objective function: triplet loss

In this chapter, we will propose a new objective function to compute the reduction matrix W, based on a triplet loss. The notion of triplet loss will be introduced in the first section where it will also be tested, then two sections will be dedicated to the enhancing of the computational performance of this new learning process. Section 5.5 presents a discussion about the computational time cost of the methods. Finally the RPA method will be combined with the triplet loss.

5.1 Triplet loss

Let consider a triplet $T_i := (X^0, X^+, X^-)$ composed by an *anchor* X^0, a *positive* X^+ and a *negative* X^-, with X^0 and X^+ being of the same class and X^- of a different class. The general idea, is to bring X^0 and X^+ closer together while taking X^- away from X^0, thanks to the optimisation process.

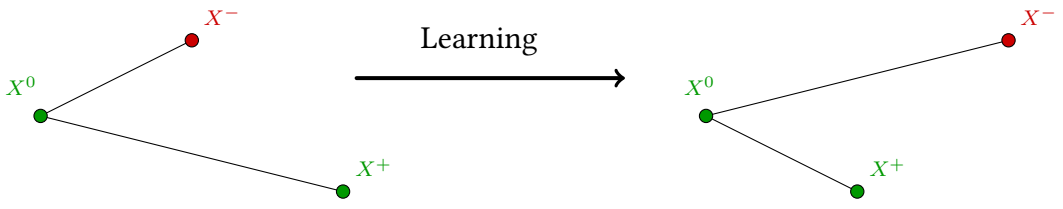

Figure 5.1: The distance between the anchor X^0 and the negative X^- is maximised, while the distance between the anchor X^0 and the positive X^+ is minimised. Based on Schroff, Kalenichenko, and Philbin (2015).

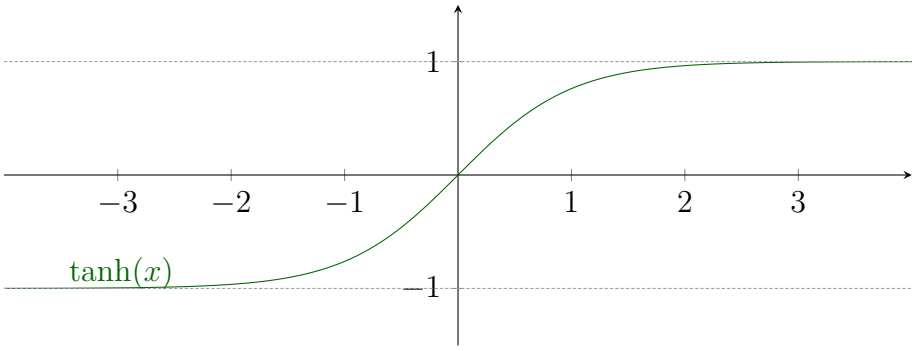

Figure 5.2: Hyperbolic tangent.

In order to achieve this goal, we will consider the following loss, for triplet T_i:

$$C_i := \tanh\left(\alpha \frac{d_p - d_n}{d_p + d_n}\right), \tag{5.1}$$

with $d_p := \delta^2(W^T X^0 W, W^T X^+ W)$, $d_n := \delta^2(W^T X^0 W, W^T X^- W)$ and $\delta^2(\cdot,\cdot)$ the square of a chosen metric (by default, the AIRM). The parameter $\alpha \in \mathbb{R}$ is a factor of 'blindness' of that cost and will be explained a bit further.

The hyperbolic tangent (see figure 5.2) has interesting properties. It is of class $\mathcal{C}^\infty$ (which is good from an optimisation point of view) and bounded by -1 and $+1$. Every point $x \in (-\infty, -3] \cup [3, \infty)$ are such that $|\tanh x| > 0.995 \approx 1$ and $|\tanh' x| < 10^{-2} \approx 0$. Therefore, a small change in x in that region is almost not perceptible. With that in mind, considering $x > 0$ and $y > 0$ we can define a characteristic factor λ such that if $x > \lambda y$, $\tanh'\left(\alpha \frac{x-y}{x+y}\right) \approx 0$ or equivalently if $x < \frac{1}{\lambda} y$, $\tanh'\left(\alpha \frac{x-y}{x+y}\right) \approx 0$, and a 'small' change in x or y will not have a big influence on the value of the function. Note that the notion of being 'small' depends on the value of x and y. If we consider that $(-\infty, -3] \cup [3, \infty)$ is an acceptable region for approximating $\tanh'$ to zero, we can compute λ as a function of α

$$\alpha \frac{x - y}{x + y} > 3 \tag{5.2}$$

$$\Leftrightarrow x > \frac{\alpha + 3}{\alpha - 3} y \tag{5.3}$$

$$\Leftrightarrow \lambda = \frac{\alpha + 3}{\alpha - 3} \tag{5.4}$$

and invert that relation to define

$$\alpha(\lambda) := 3 \frac{\lambda + 1}{\lambda - 1}. \tag{5.5}$$

Because of the above equations we require $\alpha > 3$ and $\lambda > 1$.

Going back to equation (5.1), we can understand the minimisation process of this term as follows: if $d_p > \lambda d_n$, then the anchor X^0 is further away from the positive X^+ than the negative X^- by a significant factor $\sqrt{\lambda}$. In that case, one of the three points is an outlier, and the desire of minimising C_i (linked to the derivative of $\tanh$) will be 'small'. But if $d_p < \lambda d_n$, there will be a 'strong' desire to minimise C_i, until $d_p < \frac{d_n}{\lambda}$. The advantage of using this hyperbolic tangent is thus to concentrate the efforts on interesting triplets. A quadratic function would for instance attribute a very large weight to outliers, and the final result could suffer from this.

Note that the hyperbolic tangent does not reach its lower bound so, mathematically speaking, the minimum of C_i may not be reachable. Actually, we work with iterative numerical methods, so we do not look for the perfect minimum and an 'optimal' reduction matrix will be found.

The objective function Ideally, we should consider all the possible triplets in our cost function. With four classes, sixteen matrices per class, that would make $64 \times 15 \times 48 = 46\,080$ possible triplets. This high number of triplets may give rise to a high computational burden for one evaluation of the cost function and possibly problems of convergence. We will therefore reduce that number, following the ideas of Schroff, Kalenichenko, and Philbin (2015) by considering only the triplets for which the negative is the worst possible before reduction, *i.e.*, the closest matrix of different class to the anchor point. This negative will be called the *hard* negative and the function associating an anchor point to its hard negative will be written

$$h(X^0) := \arg \min_{\substack{X \in Tr \\ c(X) \neq c(X_0)}} \delta(X, X^0)$$

with Tr the training set and $c(X)$ the class of X. Since we consider now only one possible negative per anchor point, the number of triplets to consider is now of $64 \times 15 \times 1 = 960$, which is a much more respectable number. Finally, we can write our cost function as

$$f(W; \lambda) = \sum_{\substack{X^0, X^+, X^- \in Tr \\ c(X^+) = c(X^0) \\ X^- = h(X^0)}} \tanh\left(\alpha(\lambda) \frac{d_p - d_n}{d_p + d_n}\right). \tag{5.6}$$

A discussion on the choice of parameter λ is made in section 5.4.

Computation of the Euclidean gradient As we try to minimise the objective function (5.6), it could be interesting to compute its gradient. To do this, we compute the gradient of the cost associated to one triplet. By linearity, the gradient of the whole cost function will just be the sum of the gradient of each triplet.

Considering here $N := d_p - d_n$ and $D := d_p + d_n$, the numerator and denominator of the argument of the $\tanh$ in the cost of the triplet, we can compute the Euclidean gradient as follows, by using the chain rule and the quotient rule for derivation.

$$\begin{aligned}
\operatorname{grad} C_k &= \operatorname{grad} \tanh\left(\alpha \frac{N}{D}\right) && (5.7)\\
&= \left[1 - \tanh^2\left(\alpha \frac{N}{D}\right)\right] \operatorname{grad} \alpha \frac{N}{D} && (5.8)\\
&= \left[1 - \tanh^2\left(\alpha \frac{N}{D}\right)\right] \alpha \frac{D \operatorname{grad}(N) - N \operatorname{grad}(D)}{D^2} && (5.9)\\
&= \left[1 - \tanh^2\left(\frac{N}{D}\right)\right] \alpha \frac{2d_n \operatorname{grad}(d_p) - 2d_p \operatorname{grad}(d_n)}{(d_p + d_n)^2}. && (5.10)
\end{aligned}$$

Using Yamamoto, Yger, and Chevallier (2021), we obtain

$$\operatorname{grad} d_p = 4\left(X_0 W (W^T X_0 W)^{-1} - X_i W (W^T X_i W)^{-1}\right) \log\left(W^T X_0 W (W^T X_i W)^{-1}\right) \quad (5.11)$$

$$\operatorname{grad} d_m = 4\left(X_0 W (W^T X_0 W)^{-1} - X_j W (W^T X_j W)^{-1}\right) \log\left(W^T X_0 W (W^T X_j W)^{-1}\right). \quad (5.12)$$

The implementation of this gradient has been verified using the function `checkgradient` of the `Manopt` library (see Boumal et al., 2014). The idea behind this function is to compare the value of the true cost function with its Taylor expansion. The difference between both (the residual) should behave in $\mathcal{O}(h^2)$.

More precisely, considering $\mathcal{M}$ as being the Grassmann manifold of appropriate size, $f : \mathcal{M} \to \mathbb{R}$ as being the cost function, $\operatorname{grad} f$ its Euclidean gradient and $(x, v) \in \mathrm{T}\mathcal{M}$ with $\|d\| = 1$, we can define the Taylor expansion of f around X in the direction v, with a step size h as:

$$T^f_{x,v}(h) := f(x) + h\langle \operatorname{grad} f(x), v\rangle.$$

Since we work with the Grassmann manifold, $\|\cdot\|$ refers here to the Frobenius norm and $\langle\cdot,\cdot\rangle$ to the Frobenius inner product.

Intuitively, this expansion should be compared with the value of the function f evaluated at a point $x_h \in \mathcal{M}$ that is at a distance h of x, and that lies 'in the direction of' v. To find this point, we will use the exponential map $x_h = \operatorname{Exp}_x(hv)$. The point x_h will thus be located on the maximal[1] geodesic $\gamma_v : I \to \mathcal{M}$, with $\gamma_v(0) = x$ and $\gamma'(0) = v$, at a distance h.

[1]Maximal in the sense that I is as large as possible.

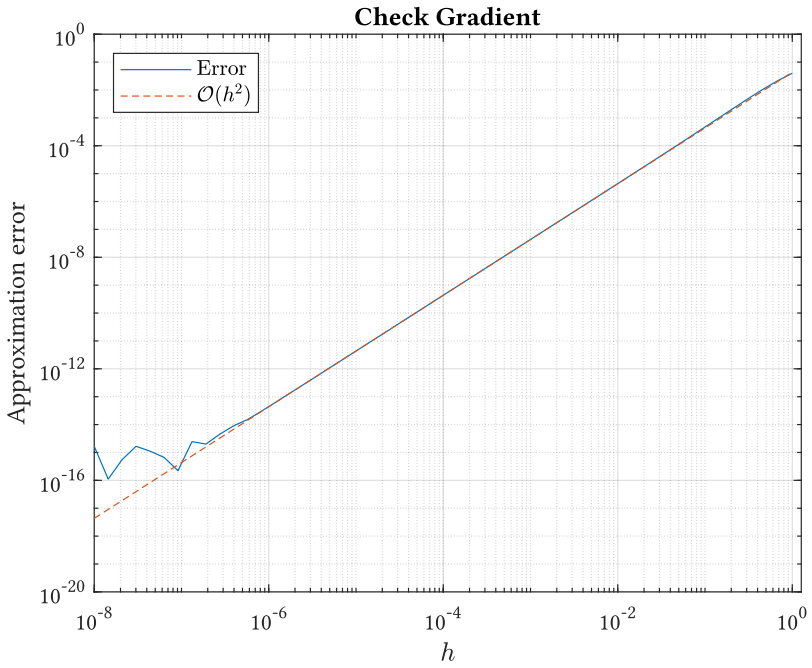

Figure 5.3: The dotted line (the residual defined in equation (5.13)) fits perfectly well the dashed one (that has a slope of 2 on this $\log - \log$ plot for small h until the residual passes below the ϵ-machine, testifying that the gradient is correct.

The residual is thus

$$R_{x,v}(h) := |f(\mathrm{Exp}_x(hv)) - T^f_{x,v}(h)| \tag{5.13}$$

$$= |f(\mathrm{Exp}_x(hv)) - f(x) - h\langle \mathrm{grad}\, f(x), v\rangle| \tag{5.14}$$

which should behave as a second order function when $h \to 0$ (see Boumal, 2020 for a detailed proof).

The pair (x, v) of the tangent bundle has been chosen at random during the procedure, such as X_0, X_i and X_j. The new dimension (width of Grassmann manifold) has been taken to 6. The result of this procedure is shown on figure 5.3. We can see that the residual is indeed in $\mathcal{O}(h^2)$ and we can conclude that the gradient computed in equations (5.7)-(5.12) is correct.

Results of numerical tests Figure 5.4 shows the pairwise distance matrix of the dataset of subject 3 before reduction and after reduction, using the whole dataset as training set for the objective function (5.6). The matrices are grouped by classes (so the first class corresponds to the lines and columns 0 to 15 and so on). One can see that after reduction, matrices of the same class are closer together (see the four blue 16×16 squares on the diagonal) and the third class seems particularly well separated from the others (there is a lot of red outside the 16×16 square on the diagonal).

To confirm that the classification happens well, a classification test has been run on the 6 first subjects and repeated three times. Per subject, ten matrices per class were chosen to become the training set and the rest as test set (same procedure as in Vaes (2020)). The results of that test is shown on figure 5.5. These results seem promising: there is an advantage to apply a reduction and the triplet method seems to perform better than Vaes' one, particularly for the MDM classifier. Note that the difference between both reductions is not very big, while the variability is relevant. We will see further that it may happen that Vaes' method produces better results.

One point worth highlighting is that the triplet loss method was slower than Vaes' one during this test. In a transfer learning perspective this is not a big issue, but in an online perspective, this is not good news. In order to enhance the computational performances of the triplet loss, two main ideas have been investigated here: working with batches and using a stochastic gradient technique. The two following sections will develop both ideas.

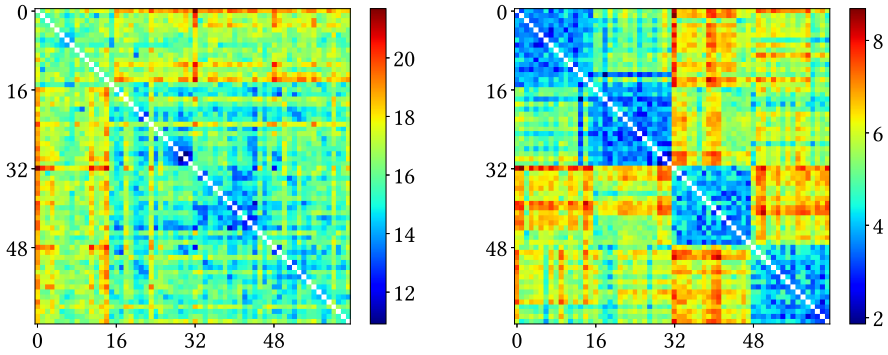

Figure 5.4: The distance matrix before (left) and after (right) reduction with the triplet method ($\lambda = 2$). We can see that each class is well separated from the others.

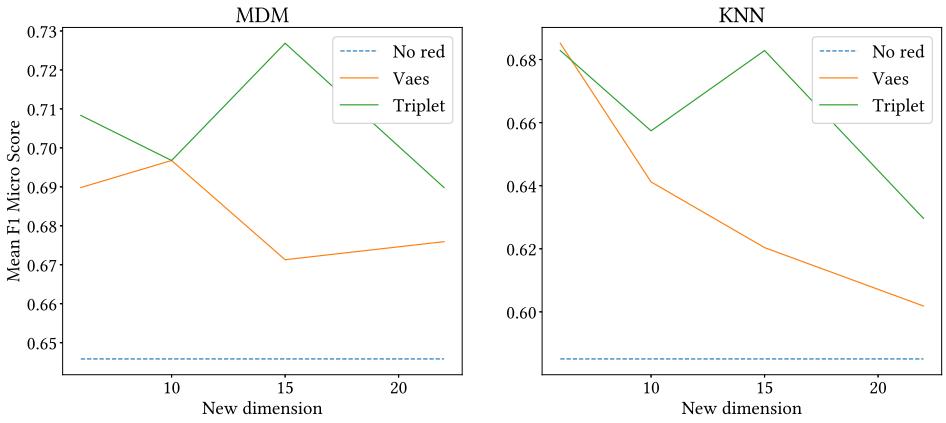

Figure 5.5: F1 Micro score for the different reduction strategy. For the triplet method $\lambda = 2$. For Vaes' method, the basic parameters have been used. The mean scores come from classifications on the 6 first subjects (training set and test generated randomly within a subject dataset), repeated 3 times. Note that a big variability has been observed.

5.2 Batches

To reduce the time needed to compute W, an approach by batches has been implemented. More precisely, the training set is divided into N_b subsets. The proportion of matrices of the same class is constant for all the sets and is thus the same of the whole training set. Using these N_b subsets as training set in the Triplet method, we obtain a set of reduction matrices $\{W_i\}_{i=1}^{N_b}$, that are then combined by computing their Riemannian barycenter:

$$W = \arg\min_{W \in \mathrm{Gr}} \sum_{i=1}^{N_b} \delta^2(W, W_i), \tag{5.15}$$

where Gr is the Grassmannian of appropriate size and δ a distance function on the Grasmmann manifold (for instance, the two-norm of the vector of principal angles between the two subspaces, such as introduced in chapter 2). Note that the optimisation problem stated in equation (5.15) might have a nonunique solution (see Afsari, 2011), because the Grassmannian can have a positive curvature (see Bendokat, Zimmermann, and Absil, 2020). However, we noticed that the produced intermediate reduction matrices are quite close to each other, hence if there exist multiple barycenters, we can hope that they are located close to each other and that they would give comparable classification performances. Moreover, if the intermediate matrices W_i are 'close enough', one can show that there is only one barycenter, even for positive curvature manifolds (Afsari, 2011). The notion of 'close enough' is not that easy to verify *a priori* in this context.

Results of numerical tests The same testing strategy as in the previous section has been applied to this technique with two batches and the results are shown on figure 5.6. We can see that the triplet loss still outperforms Vaes' one for the MDM classifier but for KNN it seems to deliver the same results.

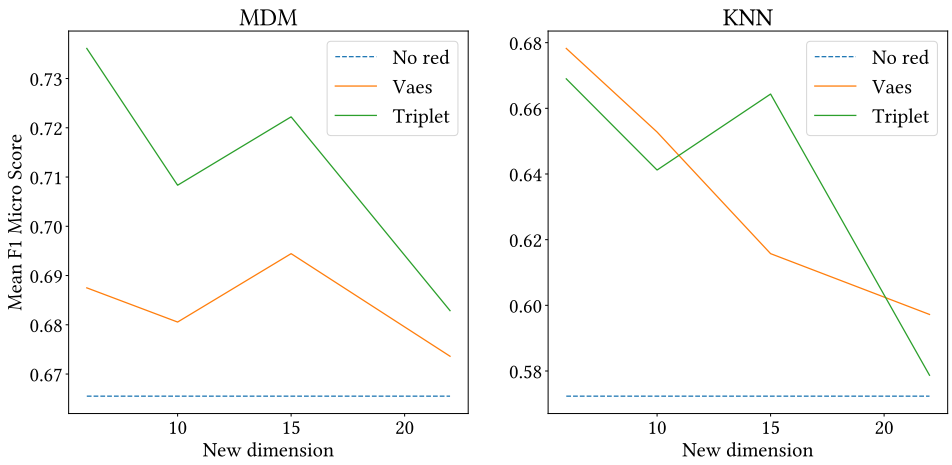

Figure 5.6: F1 Micro score for the different reduction strategy. For the triplet method $\lambda = 2$ and *two batches* have been used. For Vaes' method, the basic parameters have been used. The mean scores come from classifications on the 6 first subjects (training set and test generated randomly within a subject dataset), repeated 3 times. Note that a big variability has been observed.

5.3 Stochastic Gradient Descent

Let us now consider all the possible triplets (not only the ones with a hard negative) in the objective function. One of the reasons that would make the convergence of a minimisation algorithm on this objective function slow in time is that the gradients of all the triplets have to be computed. Actually, when the iterate is far from the optimum, we can hope that the gradients of the different terms point approximately in the same direction: the one that separates globally the sets. This setting is favourable for a *stochastic gradient descent*: choose randomly a triplet, compute its gradient and apply a Riemannian gradient descent step (see Boumal, 2020). The pseudo-code of this method is presented on algorithm 2. The gradient grad is here the Euclidean gradient and is computed in section 5.1 (see equations (5.7)-(5.12)) and $\mathrm{R}_W(s)$ designates a retraction on the Grassmannian at point W of the tangent vector s. We will use here the retraction implemented in Pymanopt (Townsend, Koep, and Weichwald, 2016).

Note that since we use the Euclidean gradient, it may happen that for some $W \in \mathrm{Gr}(D, d)$, the gradient of the cost of a triplet $\mathrm{grad}\, C(W)$ is not in the tangent space

$T_W \operatorname{Gr}(D, d)$. We therefore need to project it on the tangent space. This is done with the orthogonal[2] projector (Boumal, 2020)

$$\operatorname{Proj}_W : \mathbb{R}^{D\times d} \to T_W \operatorname{Gr}(D, d) \tag{5.16}$$

$$V \mapsto (I_D - WW^\top)V, \tag{5.17}$$

where I_D is the identity matrix of size D. The projection of the Eulidean gradient onto the tangent spaces give birth to the *Riemannian gradient* (for a submanifold).

Algorithm 2: Stochastic Gradient Descent

Pick a random $W_1 \in \operatorname{Gr}$
for $k = 1, ..., N$ **do**
  Choose randomly a triplet T_k, with associate cost $C_k(W)$, see equation (5.1)
  $\alpha_k = \frac{1}{k}$
  $s_k = -\alpha_k \operatorname{grad} C_k(W_k)$
  $W_{k+1} = \mathrm{R}_{W_k}(\operatorname{Proj}_{W_k}(s_k))$
end
return W_{N+1}

Results of numerical tests Figure 5.7 shows the pairwise distance matrix before and after reduction. Again we can see that the classes seem better separated but not as good as on picture 5.4.

The same testing strategy as previously explained has been applied to this SGD method. Figure 5.8 shows the results for N equal to 5 times the length of the training set and it is here difficult to say which method performs the best between the triplet loss and Vaes' one. Increasing N up to 25 times the length of the training set (figure 5.9) allows to enhance the performance of the SGD and we can observe that the triplet loss seems to deliver better results than Vaes' method, for the MDM classifier.

[2]With respect to the inner product on $\mathbb{R}^{D\times d}$.

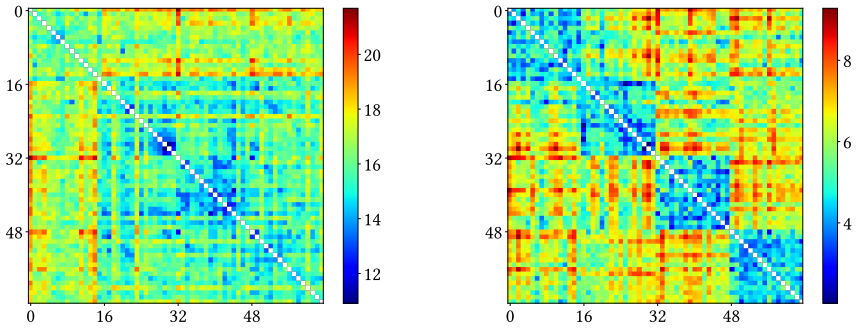

Figure 5.7: Distance matrix before (left) and after (right) reduction with the SGD approach on subject 3 ($\lambda = 2$ and $N = 640$).

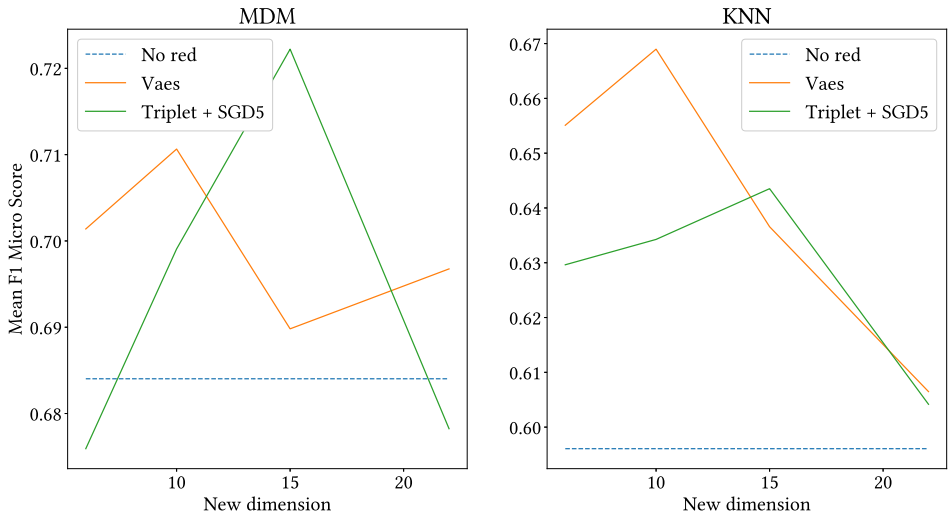

Figure 5.8: F1 Micro score for the different reduction strategy. For the SGD $\lambda = 2$ and N is set to **5** times length of the training set. For Vaes' method, the basic parameters have been used. The mean scores come from classifications on the 6 first subjects (training set and test set generated randomly within a subject dataset), repeated 3 times. Note that a big variability has been observed.

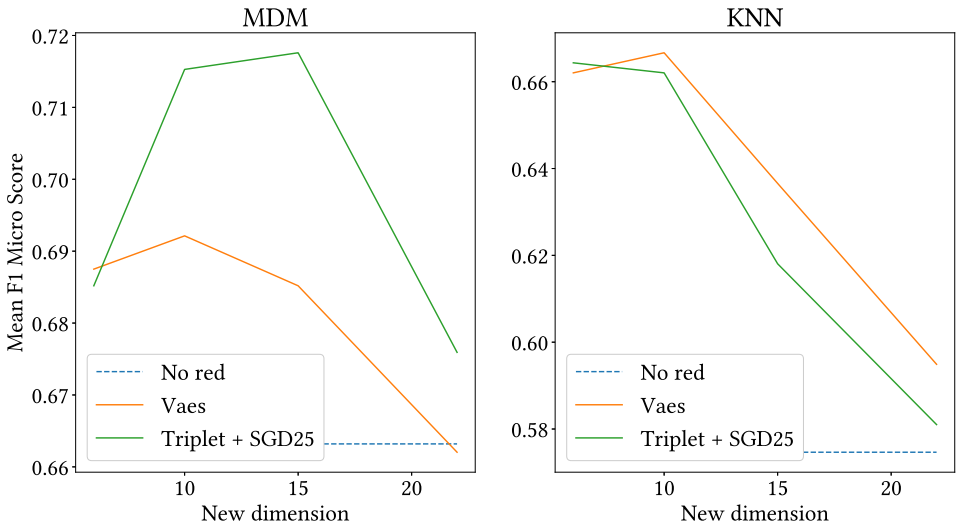

Figure 5.9: F1 Micro score for the different reduction strategy. For the SGD $\lambda = 2$ and N is set to **25** times length of the training set. For Vaes' method, the basic parameters have been used. The mean scores come from classifications on the 6 first subjects (training set and test set generated randomly within a subject dataset), repeated 3 times. Note that a big variability has been observed.

5.4 Choice of λ

In order to choose an adapted parameter λ (linked to the blindness of the objective function (see section 5.1), a test has been conducted on the 5 first subjects, for a new dimension being equal to 6, 15 and 22 and a varying λ. The optimisation method used was the SGD (with a number of steps equal to 10 times the length of training set), for computational time reasons. The results are plotted on figure 5.10. The classification score was here maximal for $\lambda = 2$. This is why in the previous sections the value of λ has been set to 2.

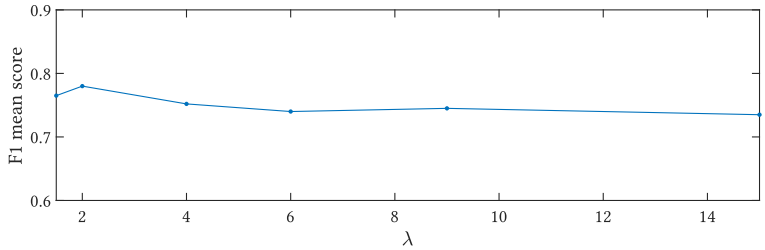

Figure 5.10: The maximal classification score is obtained when $\lambda = 2$.

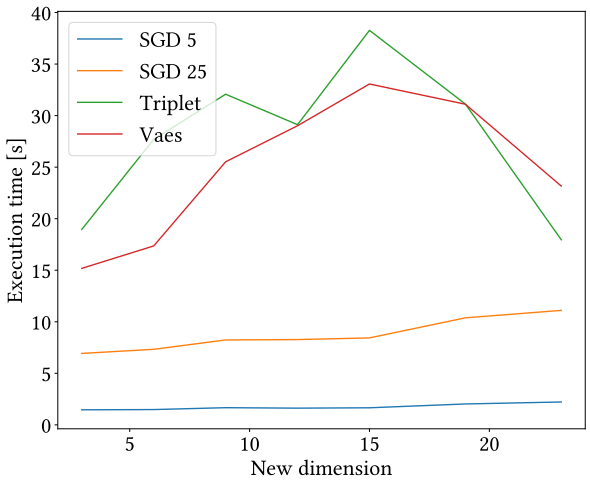

Figure 5.11: The execution time has been averaged between the 8 first subjects. The reduction for a subject is based on the 10 first matrices of each class (the training set). The 5 and 25 in the legend refer to the fact that N, the number of SGD steps, is set at 5 or 25 times the length of the training set.

5.5 Time cost

As already mentioned, the triplet loss method has been observed to be slower than Vaes' method. This is why an approach by batches, and a stochastic gradient descent have been developed. In this section we will compare the time cost of Vaes' method, the triplet method with a classical solver and the SGD approach.

Figure 5.11 presents the evolution of the time needed to find the optimal reduction matrix W with the new dimension of reduction d. We can see that the triplet method is the most expensive, then comes Vaes' method. The SGD25 and SGD5 are faster (the 25 and 5 mean that N is set to 25 or 5 times the length of the training set). We can also observe that the execution time for Vaes' method and for the triplet loss seems concave: optimising with a small or large new dimension is easy, but with an intermediate dimension it is more complicated. Concerning the small new dimension, this is probably linked to the fact that the distances in the objective are easier to compute. Concerning the $d = 23$, a deeper analysis is needed to be able to conclude anything. A first possible reason to investigate is the fact that since the dimension of the new space is quite large, there are more 'degrees of freedom', hence it is easier to converge. Note also that the SGD does not have this behaviour. This is most probably due to the fact that the number of steps is constant. We can see that the execution time for the SGD increases with the dimension. Again this is due to the fact that the reduced matrices are larger, hence the computation is more expensive.

Figure 5.12 presents the execution time of the methods with respect to the classification

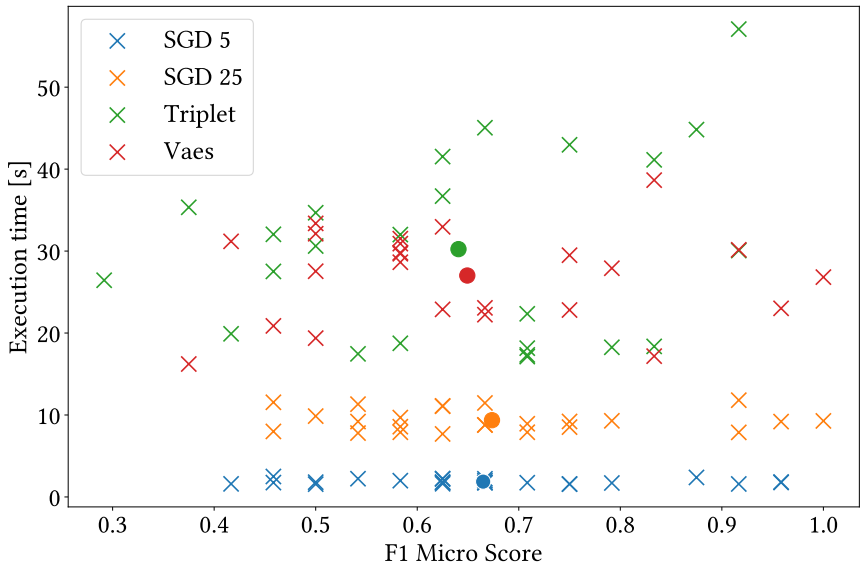

Figure 5.12: Each cross represents one reduction and the big dots are the means of the corresponding crosses. The 5 and 25 in the legend refer to the fact that N, the number of SGD steps, is set at 5 or 25 times the length of the training set.

score. For each of the 6 first subjects and a new dimension in $\{6, 10, 15, 22\}$, we built a training set with the 10 first matrices of each class, then for each of the methods we ran and timed the reduction process and evaluated the performance of the produced reduction matrix on the last matrices of the dataset (*via* a MDM classifier). Each cross on the figure represents such a test. The dots are the means of the crosses of the same colour. The perfect reduction method should produce, on such a graph, crosses near the bottom right corner: small execution time and high performances. On the opposite, a bad method would produce crosses on the 'top' left corner: high computational cost and poor performances. These are the extreme cases, but what about the intermediate cases? How can we compare two methods? Using only the means, one method is better than another if its mean lies below and on the right of the other method. If it lies below (resp. above) and on the left (resp. on the right), is it better or not? Finally, the variance/spreading of the crosses should also be taken into account. To answer such questions, choices should be made and we should develop some kind of preference function, what we will not do in this thesis.

However, a first thing that could be seen on this graph is that the triplet method seems to present a lower score than Vaes' method. As highlighted in section 5.1, where figures with the opposite conclusion are shown (see figure 5.5 for instance), these results are obtained with a certain variability and the difference in score is not significantly large enough to really conclude that one method is definitely better than the other. It

is therefore not that surprising that between two slightly different settings, the method with the highest score may change.

Another interesting point is the performance of the SGD methods. Clearly they are faster than the two other methods: comparing their mean performances, the SGD25 is 2.9 times faster than Vaes' method and SGD5 14.7 times faster. Regarding the scores, they are comparable, and even slightly better.

We did not carry out the experiment for the batch-based approach as the SGD delivers better results.

5.6 RPA and Triplet Loss

As explained previously, it could be interesting to use the data of other subjects/sessions in order to enhance the classification performance and reduce the training part that must occur before a session. This may be done by transforming a source set via RPA to make it match with a few training points of the target dataset. We applied the same learning strategy as in section 4.3.2, with the same testing procedure. The results are available in table 5.1. Vaes' reduction method still presents advantages, but the triplet method seems now worse (yet it gives at least as good results as the non reduced classification).

MDM	MDM with Vaes	MDM with Triplet	KNN	KNN with Vaes	KNN with triplet
0.71	0.74	0.71	0.63	0.68	0.66

Table 5.1: Mean F1 micro score with an RPA preprocess.

Chapter 6

Conclusion

A first contribution of this thesis has been to extend to a multi-class problem and implement in Python the DCR (Kalaganis et al., 2020). One interesting point of this technique is that it proposes a comparison criterion that can compare the quality of the reduction across dimensions. This criterion is based on the minimal spanning tree of the pairwise distance graph of the reduced matrices. The classification (with KNN or MDM) of the reduced matrices computed thanks to the DCR showed in our tests better classification performances than a classification without reduction. However the DCR was more expensive in terms of execution time.

We then worked on Vaes (2020) by investigating the case of a variable affinity matrix (we thus considered the neighbours of the reduced matrices and not of the original matrices). The classification performances were not significantly improved and the computational cost increased. The use of a constant affinity matrix is thus advocated. We then tried to see whether the produced reduction matrix W has a particular structure, in particular, does the reduction matrix select some subset of electrodes? The tests that we run did not highlight any particular pattern in the reduction matrix. Finally, we showed that a naive transfer learning approach does not work with Vaes' method: it is necessary to reconcile the foreign dataset with the current one, *via* for instance the RPA method. This requires to work in the semi-supervised paradigm.

The next chapter was dedicated to the introduction and analysis of a new objective function to compute the reduction matrix W. It is based on a triplet loss: for a triplet composed by an anchor, a positive (same class as the anchor) and a negative (different class), we bring the anchor and the positive closer together while repelling the negative from the anchor. Ideally, we should consider all the possible triplets in the optimisation process. If we do that with classical optimisation solvers, we would face computational issues, so it is necessary to reduce the number of triplets by considering only the ones that present the worst possible negative. This reduction method has shown similar classification performances to the ones obtained with Vaes' method, but the latter was faster. We thus considered a batch-based approach: the idea is to cut the training part

into smaller sets then obtain a reduction matrix for each subsets and recombine the W's. This helped to reduce the time cost, but lowered the classification performances. We therefore developed a stochastic gradient descent approach, that also allows to consider all the possible triplets: take a random triplet and make a gradient step in the direction of the gradient of that triplet. This method was, during our tests, drastically faster (between 3 or 15 times, depending on the parameters) than the initial approach and Vaes' method. Concerning the classification performances, they were comparable to the ones obtained by Vaes' method or triplet loss.

Bibliography

Afsari, B. (2011). "Riemannian L^p center of mass: existence, uniqueness, and convexity". *Proceedings of the American Mathematical Society* 139.2, pp. 655–673. URL: https://www.ams.org/journals/proc/2011-139-02/S0002-9939-2010-10541-5/S0002-9939-2010-10541-5.pdf.

Arsigny, V., P. Fillard, X. Pennec, and N. Ayache (2007). "Geometric means in a novel vector space structure on symmetric positive-definite matrices". *SIAM journal on matrix analysis and applications* 29.1, pp. 328–347.

Barachant, A. and Q. Barthélemy (2020). *pyRiemann*. URL: https://github.com/pyRiemann/pyRiemann.

Bendokat, T., R. Zimmermann, and P.-A. Absil (2020). "A Grassmann Manifold Handbook: Basic Geometry and Computational Aspects". *arXiv preprint arXiv:2011.13699*.

Beverina, F., G. Palmas, S. Silvoni, F. Piccione, S. Giove, et al. (2003). "User adaptive BCIs: SSVEP and P300 based interfaces." *PsychNology Journal* 1.4, pp. 331–354.

Botev, Z. I., D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer (2013). "The cross-entropy method for optimization". *Handbook of statistics*. Vol. 31. Elsevier, pp. 35–59.

Boumal, N., B. Mishra, P.-A. Absil, and R. Sepulchre (2014). "Manopt, a Matlab Toolbox for Optimization on Manifolds". *Journal of Machine Learning Research* 15.42, pp. 1455–1459. URL: https://www.manopt.org.

Boumal, N. (2020). *An introduction to optimization on smooth manifolds*. Available online. URL: http://www.nicolasboumal.net/book.

Bridson, M. R. and A. Haefliger (2013). *Metric spaces of non-positive curvature*. Vol. 319. Springer Science & Business Media.

Chevallier, S. (2020a). *covarianceSSVEPextraction.ipynb*. URL: https://gist.github.com/sylvchev/04b0ee8760f30da9d9c9.

— (2020b). *Dataset SSVEP Exoskeleton*. Commit faf331c. URL: https://github.com/sylvchev/dataset-ssvep-exoskeleton.

Congedo, M. (2013). "EEG source analysis". PhD thesis. Université de Grenoble. URL: https://tel.archives-ouvertes.fr/tel-00880483/document.

De Boer, P.-T., D. P. Kroese, S. Mannor, and R. Y. Rubinstein (2005). "A tutorial on the cross-entropy method". *Annals of operations research* 134.1, pp. 19–67. DOI: 10.1007/s10479-005-5724-z.

Döring, M. (2018). *Performance measures for multi-class problems*. Online, accessed on 20-05-2021. URL: https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems/.

Gower, J. C., G. B. Dijksterhuis, et al. (2004). *Procrustes problems*. Vol. 30. Oxford University Press.

Harandi, M. T., M. Salzmann, and R. Hartley (2014). "From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices". *European conference on computer vision*. Springer, pp. 17–32. DOI: 10.1007/978-3-319-10605-2_2.

Jungers, R., G. Berger, C. J., and Z. Wang (2020). *Course notes – LINMA2380 – Matrix Theory*. UCLouvain – EPL.

Kalaganis, F. P., N. A. Laskaris, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris (2020). "A Riemannian Geometry Approach to Reduced and Discriminative Covariance Estimation in Brain Computer Interfaces". *IEEE Transactions on Biomedical Engineering* 67.1, pp. 245–255. DOI: 10.1109/TBME.2019.2912066.

Kalunga, E. K., S. Chevallier, O. Rabreau, and E. Monacelli (2014). "Hybrid interface: Integrating BCI in multimodal human-machine interfaces". *2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 530–535. DOI: 10.1109/AIM.2014.6878132.

Kamarajan, C. et al. (2015). "Deficient event-related theta oscillations in individuals at risk for alcoholism: a study of reward processing and impulsivity features - Fig 2". *PloS one* 10.11, e0142659. DOI: 10.1371/journal.pone.0142659.g002.

Lafon, S. and A. B. Lee (2006). "Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization". *IEEE transactions on pattern analysis and machine intelligence* 28.9, pp. 1393–1403.

Ledoit, O. and M. Wolf (2004). "A well-conditioned estimator for large-dimensional covariance matrices". *Journal of multivariate analysis* 88.2, pp. 365–411.

Lee, J. M. (2018). *Introduction to Riemannian manifolds*. Springer. DOI: 10.1007/978-3-319-91755-9.

Lotte, F. et al. (2018). "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update". *Journal of neural engineering* 15.3, p. 031005.

Mishra, A. (2018). *Metrics to evaluate your machine learning algorithm*. Online, accessed on 20-05-2021. URL: https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234.

Nielsen, F. (2016). *Stein divergence – Properties*. Laboratoire d'informatique de l'École polytechnique. URL: https://www.lix.polytechnique.fr/~nielsen/SD-SteinDivergence.pdf.

Nunez, P. L. and R. Srinivasan (2007). "Electroencephalogram". *Scholarpedia* 2.2. revision #91219, p. 1348. DOI: 10.4249/scholarpedia.1348.

Rodrigues, P. L. C., C. Jutten, and M. Congedo (2019). "Riemannian Procrustes Analysis: Transfer Learning for Brain–Computer Interfaces". *IEEE Transactions on Biomedical Engineering* 66.8, pp. 2390–2401. DOI: 10.1109/TBME.2018.2889705.

Rubinstein, R. (1999). "The cross-entropy method for combinatorial and continuous optimization". *Methodology and computing in applied probability* 1.2, pp. 127–190. DOI: 10.1023/A:1010091220143.

Schroff, F., D. Kalenichenko, and J. Philbin (2015). "FaceNet: A Unified Embedding for Face Recognition and Clustering". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Schroff_FaceNet_A_Unified_2015_CVPR_paper.pdf.

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

Townsend, J., N. Koep, and S. Weichwald (2016). "Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation". *Journal of Machine Learning Research* 17.137, pp. 1–5. URL: http://jmlr.org/papers/v17/16-177.html.

Tu, L. (2010). *An Introduction to Manifolds*. Universitext. Springer New York. DOI: 10.1007/978-0-387-48101-2.

Vaes, C. (2020). "Classification of biological signals on the symmetric positive definite manifold". Prom. : P.-A. Absil, E. Massart. MA thesis. École Polytechnique de Louvain, Université Catholique de Louvain. URL: http://hdl.handle.net/2078.1/thesis:25234.

Vallabhaneni, A., T. Wang, and B. He (2005). "Brain—computer interface". *Neural engineering*. Springer, pp. 85–121.

Wong, Y.-C. (1967). "Differential geometry of Grassmann manifolds". *Proceedings of the National Academy of Sciences* 57.3, pp. 589–594. DOI: 10.1073/pnas.57.3.589. URL: https://www.pnas.org/content/57/3/589.

Yamamoto, M. S., F. Yger, and S. Chevallier (June 2021). "Subspace oddity - optimization on product of Stiefel manifolds for EEG data". *ICASSP*. Toronto, Canada. URL: https://hal.archives-ouvertes.fr/hal-03202357.

Yger, F., M. Berar, and F. Lotte (2017). "Riemannian Approaches in Brain-Computer Interfaces: A Review". *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.10, pp. 1753–1762. DOI: 10.1109/TNSRE.2016.2627016.

Appendix A

Default parameters

Table A.1 contains the default values of the parameters used in this thesis.

Training set	
size	10 matrices per class
KNN classifier	
k	$\min(5, n_{m/c})$
Vaes's method	
v_b	3
v_w	$n_{m/c}$
Triplet	
λ	2
DCR (see Kalaganis et al., 2020, p. 249 for a definition of the parameters)	
ρ	0.5
N	15
Max. # iter.	1000
Miscellaneous	
Metric on $\mathcal{S}_{++}^n$	AIRM

Table A.1: Default parameters. The minimal number of matrix/point per class in the training set is denoted $n_{m/c}$.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**