UCLouvain

École polytechnique de Louvain

Classification of covariance matrices for EEG: How to handle the low-rank case?

Author: **Diane GAILLY**
Supervisors: **Pierre-Antoine ABSIL, Estelle MASSART**
Reader: **Noémie JAQUIER**
Academic year 2021–2022
Master [120] in Mathematical Engineering

Remerciements

Je tiens à remercier très chaleureusement toutes les personnes ayant contribué à ce mémoire.

Je voudrais dans un premier temps sincèrement remercier mes promoteurs, le Professeur Pierre-Antoine Absil et la Docteure Estelle Massart pour leur encadrement éclairant, leur intérêt insatiable et leur judicieux conseils tout au long de ce travail. La confiance qu'ils m'ont accordée m'a vraiment touchée et motivée à avancer.

Je remercie également la Docteure Noémie Jaquier d'avoir si gentiment accepté de faire partie de mon jury et de m'accorder de son temps pour évaluer mon mémoire avec beaucoup d'enthousiasme.

Enfin, je tiens à remercier mes parents et ma sœur, qui ont toujours été là pour moi, pour leur soutien inconditionnel et leurs encouragements. Merci aussi à Mattéo et à sa famille pour leur présence précieuse et réconfortante, aussi bien dans les bons que dans les mauvais moments.

Abstract

Brain-Computer Interfaces (BCIs) allow humans to communicate with a device solely via brain activity. Most BCIs rely on the classification of electroencephalography (EEG) signals to translate brain activity into commands. Existing state-of-the-art classification methods, based on Riemannian geometry, show their limits when the covariance matrices used to represent the signals become ill-conditioned. In this work, we consider a new Riemannian metric, the fixed-rank Wasserstein metric, which allows to take this low-rank structure into account. We compare this new metric with two other metrics, the Euclidean metric and the Affine-Invariant Riemannian metric, via two distance-based classification methods, Minimum Distance to the Mean and k-Nearest Neighbors. We also evaluate the impact of shrunk covariance estimators which are common estimators for alleviating the effect of ill-conditioning. Our results show that the Wasserstein metric achieves similar classification performance to the affine-invariant metric and uses less computation time and memory resources if the rank is wisely chosen. We also show that the new metric is practically independent of the matrix shrinkage and hence does not suffer from ill-conditioning. The Wasserstein metric is therefore of great interest for high-dimensional EEG signals and could outperform state-of-the-art Riemannian approaches in BCI.

Contents

Chapter 1

Introduction

1.1 Motivation and context

Who has never dreamed of being able to communicate with others only by thought or of being able to control his television without lifting a finger? Although these ideas may seem crazy and unattainable, this might not be the case anymore in a few years... At least that's what we hope!

For a long time, the idea of using the brain to communicate or perform control actions has been of great interest to scientists. This has given rise to the field of Brain-Computer Interfaces (BCIs) [WW12]. As the name indicates, BCIs are systems that allow their users to transmit information to a device only using the brain signals they generate. The device will then allow the users to interact with their internal and/or external environment. These brain signals can be recorded in several different ways. The most used method is electroencephalography (EEG), which measures the electrical activity of the brain using electrodes placed on the scalp [LBC15].

Initially, BCIs were developed to help people with severe disabilities. It is still mainly in this spirit that BCIs are developed now. Through their different applications, BCIs allow to replace, restore and improve a lost or deficient function.

Replace With the help of BCIs, users who have lost abilities such as speech or the use of limbs can replace them. Spellers, for example, allow users to write sentences without using muscle control to communicate with their surroundings; controlled wheelchairs help their users with mobility issues to get around.

Restore It is also possible to restore lost abilities by stimulating the paralyzed muscle or nerve and thus generate the desired movement or action. These paralyses can occur following an accident where the spinal cord has been affected or following a disease such as multiple sclerosis which can leave disorders. For example, BCI may enable urination control in the case of bladder function problems.

Improve Following a stroke, certain movements, particularly those of the arms and hands, may be impaired. BCIs can be used to rehabilitate the user's brain to remodel their connections according to the environment and the experiences they are having in order to regain (partial) motor control. This ability of the brain is called "brain plasticity".

More and more applications are also aimed at healthy users in order to improve or contribute to certain tasks, whether they are of a useful or recreational nature. In these cases we will say that the BCIs enhance and/or supplement well functioning functions.

Enhance In many scenarios, a person's attention is crucial. Some applications propose to monitor this attention in order to prevent inattention errors: suggest a break to the user whose vigilance decreases during work requiring great attention (i.e. security check); avoid a car accident by detecting the user's braking signals and applying the car's brakes faster than the user.

Supplement BCIs can also be used in a playful way in video games and virtual realities, as an original means of control. One example is the power 4 game where the user selects the column in which he wants to put his token. Or the video game in which the BCI is used in a complementary way with a traditional control system in order to pick up objects by staring at them when they pass by.

Thanks to their numerous applications and their multidisciplinary nature, BCIs constitute a field that is of great interest to many researchers from different backgrounds. Moreover, over the last thirty years, the field has experienced impressive growth due to numerous advances: technological such as the development of more powerful and less expensive computers, scientific with a deeper understanding of the human body and societal with a better understanding of the needs of people with disabilities and the recognition of the benefits that BCIs bring to their daily lives [NNL18].

When creating a BCI system, the first issue is to process the brain signals. For this, different mathematical concepts are used, the most central of which is the notion of covariance matrix which allows us to differentiate and therefore also compare the signals between them.

In a second step, we have to recognize what the user wants to do with the help of classification methods. Recently, new methods based on Riemannian geometry have emerged. The fact that they take into account the intrinsic nature of covariance matrices allows to reach better classification performance. These methods are therefore very promising and could soon be considered as the new state of the art in BCI [YBL17].

In the meantime, a new generation of EEG, called High-density electroencephalography (HD-EEG) has been developed to overcome the major drawback of traditional EEG: the lack of spatial resolution. For this purpose, these HD-EEGs can have up to 256 electrodes, much more than traditional EEGs [Pis+14].

Since the covariance matrices represent the signals, if the signals are recorded with a larger number of electrodes, the size of the covariance matrices grows proportionally. When these covariance matrices become larger, a low-rank structure naturally appears. However, the majority of the existing Riemannian methods rely on the fact that the matrices are of full-rank, which is not the case anymore with a large number of electrodes. For this reason, these Riemannian approaches no longer seem to be the right choice in this case.

1.2 Aim of the work

A classical way to represent EEG signals in order to classify them is to use their covariance matrices. By nature, covariance matrices are Symmetric Positive Definite (SPD) matrices. The set of these SPD matrices defines a particular space: the SPD manifold. On this space, an appropriate metric has been defined to take into account the intrinsic geometry of the space. This metric, called Riemannian metric, allows us to define appropriate distances on this space.

When these matrices become large, a low-rank structure appears. These new matrices, called rank deficient, are no longer SPD and therefore no longer belong to the SPD manifold.

The aim of this work is to explore a metric which seems appropriate to take the low-rank nature of covariance matrices into account: the Wasserstein metric in its low-rank form.

For this purpose, we will compare the state of the art in EEG signal classification with this new metric, based on performance measurements. The state of the art includes approaches based on Riemannian geometry, which take the structure of the space into account by using adapted Riemannian metrics and corresponding distances.

1.3 Structure of the work

The thesis is structured as follows:

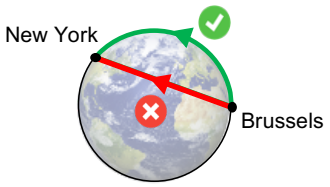
- In chapter 2, we introduce all the mathematical notions necessary for the understanding of this thesis. First, we will formally define what a manifold is. Then, we will define the metrics and means that we will use.
- In chapter 3, we will focus on Brain-Computer Interfaces (BCIs). We will briefly introduce some basic notions in neuroscience. Then, we will talk about the two families of approaches for EEG-based BCIs, the "classical" approaches and the Riemannian approaches.
- In chapter 4, we will explain first the basic principles of the two classifiers we will use, then the different existing performance measures and finally the model validation techniques used to compare the classifiers.
- In chapter 5, we present the dataset that we experimented on as well as the different classification pipelines. We lay out the conducted experiments, and finally outline our results and discuss them.
- In chapter 6, we will conclude the thesis by proposing various improvement suggestions and also ideas for future research paths.

Chapter 2

Theory of Manifolds

Our goal is to classify EEG signals. To do that, we will use *distance-based classification methods*. However, we must be very careful when using this notion of distance.

Let us consider the following analogy: we are flying and we want to know the distance between Brussels and New York. We can first calculate the shortest distance in space. But this distance, represented by the red line, crosses the earth, which is impossible. To have a more realistic distance we must consider the space in which the two cities are located: the surface of the earth. Since the earth is a sphere, we have to take into account its curvature to measure the distance between the two cities accurately. This can be done by using the green curve instead of the red line. To measure a distance, it is essential to first define the space in which it is located.



Let us go back to EEG-based BCIs. Since we are using covariance matrices to represent EEG signals, we first need to define the space in which these covariance matrices lie in order to be able to define the notion of distance in this space. By essence, covariance matrices are Symmetric and Positive Definite (SPD). They therefore live in the space of $n \times n$ SPD matrices, denoted $\mathcal{S}_n^{++}$; this is a special space called *manifold*.

The goal of this chapter is to introduce the mathematical concepts necessary for the understanding of the thesis. First, we define the concept of manifold formally. Then, we explore the manifold of symmetric and positive definite (SPD) matrices where the covariance matrices live. We define the different Riemannian metrics that we exploit on this manifold, as well as the methods used to compute means of SPD matrices according to these metrics. Finally, we define the low-rank approximations of SPD matrices which live in a special manifold; on this manifold we define the new Riemannian metric which is the basis of this work.

2.1 Introduction to manifolds

First of all, after a brief introduction to topology notions, we define what is a (n-dimensional) *topological manifold* which is the basis of our reflection. Intuitively, a *manifold* is a topological space that *locally* looks like a Euclidean space. Moreover, in order to be able to define tools to measure distances, this topological manifold needs to be *smooth*, i.e., possess a differentiable structure. Then, we can introduce the *metric tensor* that allows us to compute distances. This metric tensor and the smooth manifold form a new manifold called the *Riemannian manifold*.

A more comprehensive introduction to Riemannian manifolds can be found in [AMS09; Lee18].

2.1.1 Notions of topology

We begin by defining the very general mathematical concept of topological space.

Definition 1 (Topological space)**.** A topological space is a set of "points" X along with a set of subsets (or *neighborhoods*) $\mathcal{N}(x)$ for each point $x \in X$. The neighborhoods must verify a set of axioms; we do not detail these to keep an intuitive overview.

A concrete example of topological space is the real line $\mathbb{R}$, where the neighborhoods of point x are all the open intervals containing x. This simple definition is already sufficient to define the notion of *closeness*: a point $x \in X$ is *close* to a set $S \subset X$, either if x belongs to S, or every neighborhood of x intersects with S (in the latter case, we say that x is a *limit point* of S).

Before moving on to manifolds, we need to define two technical concepts that will prevent pathological cases to be defined as manifolds.

Definition 2 (Hausdorff space)**.** A topological space is a *Hausdorff space*, or a *separated space*, if any two distinct points x and y admit neighborhoods $N \in \mathcal{N}(x)$ and $M \in \mathcal{N}(y)$ which are disjoint.

Hausdorffness of topological spaces is quite natural; we do not present non-examples as it would require higher levels of abstraction. The second concept first requires to define *open sets*[1]: a subset $U \subset X$ is said to be *open* if U is a neighborhood of every point in U.

Definition 3 (Basis of a topological space)**.** A basis of a topological space X is a family of open sets of X such that every open set of X can be written as the union of elements of the base.

2.1.2 Topological manifolds

Previously, we said that a manifold is a space that locally "looks like" a Euclidean space. Let us define what "looking like" formally means; first, we define homeomorphisms, which are functions that express a kind of equivalence between two topological spaces:

[1] There is an equivalence between open subsets and neighborhoods of topological spaces. The standard definition of topological space actually involves open sets instead of neighborhoods, but we gave the neighborhood definition as it is more intuitive.

Figure 2.1: A mug is homeomorphic to a torus, since there exists a continuous mapping from one to another.

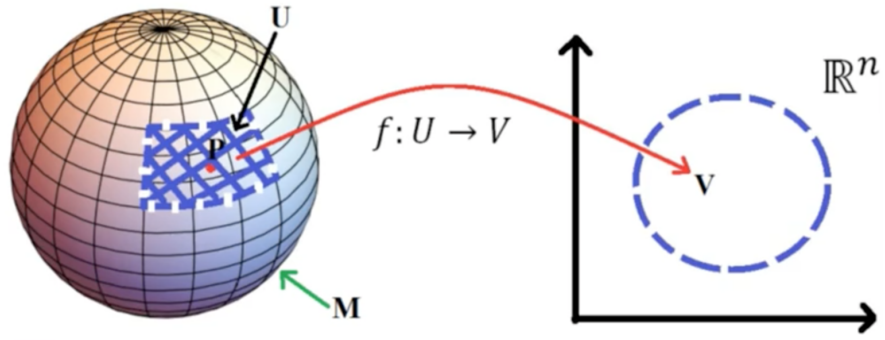


Figure 2.2: Illustration of a chart f mapping a neighborhood U of P onto a Euclidean subset V. From [Dav16].

Definition 4 (Homeomorphism)**.** A homeomorphism is a continuous function that maps one topological space to another and has a continuous inverse function.

If two topological spaces admit a homeomorphism mapping one to another, they are *homeomorphic*. Homeomorphism is the formal expression of "looking like". See Fig. 2.1 for an illustration of homeomorphic spaces.

We finally have all the tools to define topological manifolds:

Definition 5 (Topological manifold)**.** An n-dimensional topological manifold $\mathcal{M}$ a Hausdorff space with a countable basis and which is locally homeomorphic to $\mathbb{R}^n$: for every point x on $\mathcal{M}$ there is a neighborhood U of x and a homeomorphism $f : U \to V$ which maps U to a subset $V \subset \mathbb{R}^n$.

The homeomorphisms $f : U \to V$ mapping neighborhoods to Euclidean subsets are called *charts* (see Fig. 2.2), and a collection of charts $\{f_\alpha : U_\alpha \to V_\alpha\}$ is called an *atlas* if the charts cover the manifold completely: $\cup_\alpha U_\alpha = \mathcal{M}$. Informally, an atlas is a mapping from the manifold to a "piecewise Euclidean" space. A *maximal atlas* is one that cannot admit additional charts.

A natural question that may arise is the following: when two neighborhoods U_α, U_β overlap, what happens in the Euclidean space? This is illustrated in Fig. 2.3 and leads us to the definition of coordinate transformation.

Definition 6 (Coordinate transformation)**.** If f_α and f_β are two charts, then on the intersection of their domains $U_\alpha \cap U_\beta$ there exists another homeomorphism $f_{\alpha\beta} : f_\alpha(U_\alpha \cap$

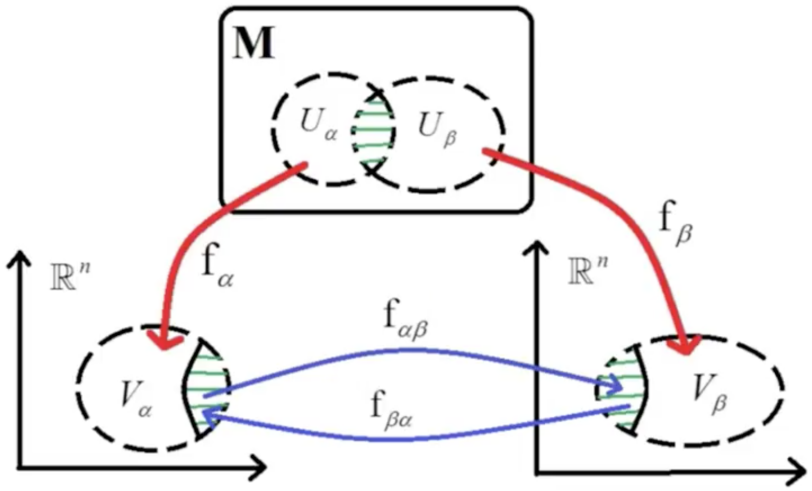


Figure 2.3: Illustration of overlapping charts f_α, f_β and their coordinate transformation $f_{\alpha\beta}$. From [Dav16].

$U_\beta) \to f_\beta(U_\alpha \cap U_\beta)$ where $f_{\alpha\beta} = f_\beta \circ f_\alpha^{-1}$. This mapping gives a relation between the coordinates in the two charts and is called a coordinate transformation or just a change of coordinates.

These coordinate transformations are important because they allow us to define new classes of differentiable manifolds. By differentiable we mean that the derivatives of k-orders exist (class C^k) and are continuous on $\mathbb{R}^n$. From transformations that are infinitely differentiable (class C^∞) we can define smooth manifolds.

2.1.3 Smooth manifold

Definition 7 (C^∞ altas)**.** An atlas defined on a manifold is said to be C^∞ if all of its coordinate transformations are infinitely differentiable mappings.

Definition 8 (C^∞ structure)**.** A maximal C^∞ atlas on a topological manifold is a C^∞ structure.

Definition 9 (Smooth manifold)**.** A smooth manifold is a topological manifold together with a C^∞ structure.

Now that we have defined the smooth manifold, we can embed it with a metric in order to compute distances on it, and obtain a *Riemannian manifold*. We will proceed in several steps. First of all we will define intuitively the concepts of *tangent vector* and *tangent space*, which are necessary for the definition of a metric. Then, we will present different existing metrics and their associated distances.

2.1.4 Riemannian manifold

The notion of tangent vector at a point x on a manifold $\mathcal{M}$ can be defined using the notion of curve (see Fig. 2.4). Let $\gamma : \mathbb{R} \to \mathcal{M}$ be a parametric curve passing through x

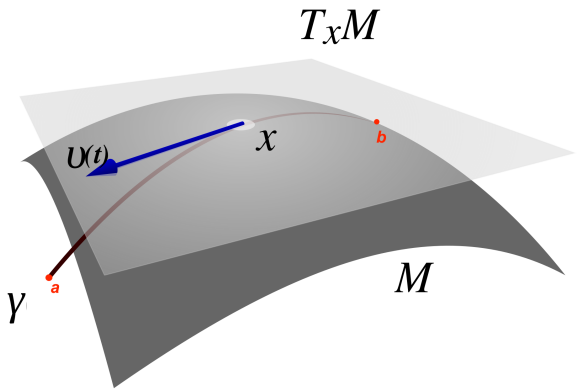


Figure 2.4: Tangent vector $v(t)$ and tangent space T_xM representation on the manifold $\mathcal{M}$ at a point x of the curve γ. Adapted from [Wik21].

at "time" t: $\gamma(t) = x$. The velocity vector $v(t) = \mathrm{d}\gamma(t)/\mathrm{d}t$ of the curve at point x is a *tangent vector*. The tangent vectors corresponding to all curves passing through x make up the *tangent space* at x, denoted $T_x\mathcal{M}$.

We have given an intuitive definition of the tangent vector. Now we want to define its length. We can do this by defining an inner product on the tangent space, called a *metric*. Formally, a metric is a bilinear, symmetric positive-definite form on the tangent space $T_x\mathcal{M}$, denoted $\langle\cdot,\cdot\rangle_x$. From this inner product we can measure the norm a tangent vector $v \in T_x\mathcal{M}$:

$$\|v\|_x = \sqrt{\langle v, v\rangle_x}. \tag{2.1}$$

Definition 10 (Riemannian metric)**.** A Riemannian metric is a smoothly varying inner product. This inner product is defined on the tangent space $T_x\mathcal{M}$ of a manifold $\mathcal{M}$.

Definition 11 (Riemannian manifold)**.** A Riemannian manifold defines a smooth manifold endowed with a Riemannian metric. Formally, the couple $(\mathcal{M}, g)$, where $\mathcal{M}$ is a smooth manifold and g is a Riemannian metric on $\mathcal{M}$, is a Riemannian manifold.

Let us reconsider Fig. 2.4 and imagine that we want to calculate the length of curve γ between two points a and b. We have nothing that allows us to calculate this distance on this manifold. However, we do know how to measure lengths on the tangent space at a point of this manifold. Knowing this, we can partition the curve between a and b into very small sections which are almost line segments. We can therefore use the Riemannian metric defined on the tangent space to compute their lengths. By adding them together, we will obtain the total length of the curve between points a and b. Let t_a, t_b denote the parameters of γ corresponding to points a, b. We have

$$\begin{aligned} L(\gamma) &= \int_{t_a}^{t_b} \|\mathrm{d}\gamma\,(t)\| \\ &= \int_{t_a}^{t_b} \|v(t)\|\,\mathrm{d}t && \text{since } v(t) = \frac{\mathrm{d}\gamma}{\mathrm{d}t} \\ &= \int_{t_a}^{t_b} \sqrt{\langle v(t), v(t)\rangle_{\gamma(t)}}\,\mathrm{d}t && \text{by (2.1)} \end{aligned}$$

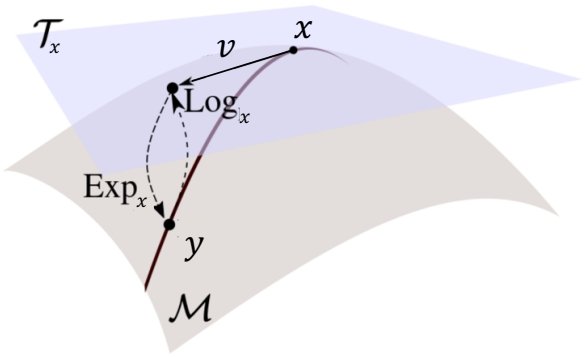


Figure 2.5: Exponential and logarithmic maps, projecting the point y on the vector v tangent to x, and vice-versa. Adapted from [Sab+19].

Being able to measure curve lengths allows to define the distance function induced by a Riemannian metric: the distance between points a and b is the length of the shortest curve joining a and b:

$$\delta(a, b) = \min_{\gamma: a \to b} L(\gamma) \tag{2.2}$$

Such a shortest curve is called a *geodesic*.

Exponential and logarithmic maps For many applications it is useful to be able to "project" a tangent vector on the manifold, and conversely, to project a point of the manifold on some tangent space. Such projections, depicted in Fig. 2.5, are called the *exponential* and *logarithmic maps*; respectively:

$$\mathrm{Exp}_x : T_x\mathcal{M} \to \mathcal{M}, \qquad \mathrm{Log}_x : \mathcal{M} \to T_x\mathcal{M}.$$

Given a tangent vector $v \in T_x\mathcal{M}$, there is a unique geodesic γ_v such that $\gamma_v(0) = x$ and $\gamma_v'(0) = v$ (γ_v starts at x in direction v). Then the exponential map is defined by $\mathrm{Exp}_x(v) = \gamma_v(1)$. Conversely, given a point $y \in \mathcal{M}$, the logarithmic map is defined as the tangent vector of the unique geodesic $\gamma_{x,y}$ starting at x and joining y: $\mathrm{Log}_x(y) = \gamma_{x,y}'(0)$.

2.2 Manifold of symmetric positive definite matrices

Since we work with SPD matrices, we are interested in the space in which they live: the SPD manifold. This manifold belongs to the family of Riemannian manifolds, that is to say that it benefits from all the computational properties that we have developed above with the difference that the elements that compose it are SPD matrices. Knowing that, we will define this manifold as well as the metrics and distances associated with it.

Definition 12 (Symmetric positive definite (SPD) matrix)**.** A real $n \times n$ matrix $\mathbf{X}$ is symmetric positive definite if it is:

- symmetric: $\mathbf{A} = \mathbf{A}^\top$,
- positive definite: $\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$, for all nonzero vectors $\mathbf{x}$.

Definition 13 (SPD manifold)**.** Symmetric positive definite matrices of size $n \times n$ form a Riemannian manifold, denotes $\mathcal{S}_n^{++}$.

In what follows, we will regularly use the notation $\mathbf{X}^{1/2}$ which corresponds to the square root of a positive definite matrix.

Definition 14 (Square root of a symmetric positive definite matrix)**.** The square root of a symmetric positive definite matrix $\mathbf{X}$ is the unique symmetric positive definite matrix $\mathbf{Y}$ such that $\mathbf{X} = \mathbf{Y}^2$; it is noted $\mathbf{X}^{1/2}$. It is simply computed via the eigendecomposition of $\mathbf{X}$: if $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, with $\mathbf{U}$ orthogonal and $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, then the matrix $\mathbf{Y} = \mathbf{U}\mathbf{\Lambda}^{1/2}\mathbf{U}^\top$, with $\mathbf{\Lambda}^{1/2} = \operatorname{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n}\right)$ is the square root of $\mathbf{X}$; indeed, $\mathbf{Y}$ is positive definite and $\mathbf{Y}^2 = \mathbf{X}$.

We will also often note $\mathbf{X}^{-1/2}$ which is the inverse of $\mathbf{X}^{1/2}$. We can also define the *matrix exponential* and *matrix logarithm*:

Definition 15 (Matrix exponential and logarithm of SPD matrix)**.** Let $\mathbf{X} \in \mathcal{S}_n^{++}$ be a SPD $n \times n$ matrix, and $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ its eigendecomposition. The matrix exponential expm and logarithm logm are given by

$$\operatorname{expm}(\mathbf{X}) = \mathbf{U} \operatorname{diag}\left(e^{\lambda_1}, \ldots, e^{\lambda_n}\right)\mathbf{U}^\top, \qquad \operatorname{logm}(\mathbf{X}) = \mathbf{U} \operatorname{diag}\left(\log \lambda_1, \ldots, \log \lambda_n\right)\mathbf{U}^\top.$$

The matrix exponential and logarithm possess properties similar to the exponential and logarithm of real numbers. In particular, notice that $\operatorname{expm}(\operatorname{logm}(\mathbf{X})) = \mathbf{X}$.

2.2.1 Metrics for the manifold of symmetric positive definite matrices

The objective of this work being to classify covariance matrices, we need to measure distances between these symmetric positive definite matrices. For this, we need to embed the manifold with a metric $\langle \cdot, \cdot \rangle_{\mathbf{X}}$ inducing a distance $\delta(\cdot, \cdot)$. We describe here the Euclidean metric as well as two Riemannian metrics well known in the literature: the *affine-invariant Riemannian metric* [Moa05] and the *Bures-Wasserstein metric* [BJL19]. In what follows, $\mathbf{X}, \mathbf{S}_1, \mathbf{S}_2$ are SDP matrices and $\mathbf{A}, \mathbf{B}$ are symmetric matrices (belonging to the tangent space of the manifold).

Euclidean metric The simplest metric treats matrices as if they live in the Euclidean space of symmetric matrices. The Euclidean metric is then the Frobenius scalar product:

$$\langle \mathbf{A}, \mathbf{B} \rangle_{\mathbf{X}} = \operatorname{tr}\left(\mathbf{A}^\top \mathbf{B}\right).$$

Note that this metric does not depend on the anchor point $\mathbf{X}$, which shows that it does not take into account the structure of the manifold. The distance induced by this metric is the Frobenius distance:

$$d(\mathbf{S}_1, \mathbf{S}_2) = \|\mathbf{S}_1 - \mathbf{S}_2\|_{\mathrm{F}}. \tag{2.3}$$

Affine-invariant Riemannian metric A particularly popular choice is the affine-invariant Riemannian metric (AIRM), defined on the tangent space as

$$\langle \mathbf{S}_1, \mathbf{S}_2 \rangle_{\mathbf{X}} = \operatorname{tr}\left(\mathbf{X}^{-1}\mathbf{S}_1\mathbf{X}^{-1}\mathbf{S}_2\right).$$

This metric induces the distance

$$\delta(\mathbf{S}_1, \mathbf{S}_2) = \left\|\operatorname{logm}\left(\mathbf{S}_1^{-1}\mathbf{S}_2\right)\right\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n} \log^2 \lambda_i}, \tag{2.4}$$

where $\operatorname{logm}(\cdot)$ is the matrix logarithm, and λ_i, $i, = 1, \ldots, n$ are the eigenvalues of $\mathbf{S}_1^{-1}\mathbf{S}_2$. Note that, due to the inversion, this metric is not well-suited if the matrices are ill-conditioned, i.e., $\lambda_{\max}/\lambda_{\min}$ is large.

This metric is called *affine-invariant* because is it invariant under an affine transformation of $\mathbf{S}_1, \mathbf{S}_2$, i.e., for any invertible matrix $\mathbf{W}$,

$$\delta\left(\mathbf{W}^\top\mathbf{S}_1\mathbf{W}, \mathbf{W}^\top\mathbf{S}_2\mathbf{W}\right) = \delta(\mathbf{S}_1, \mathbf{S}_2).$$

Bures-Wasserstein metric This metric is particularly interesting because it has an extension for positive semi-definite matrices of fixed rank. We do not detail the inner product on the tangent space. The distance between two positive semi-definite matrices is given by [BJL19]

$$\delta(\mathbf{S}_1, \mathbf{S}_2) = \sqrt{\operatorname{tr}(\mathbf{S}_1) + \operatorname{tr}(\mathbf{S}_2) - 2\operatorname{tr}\left(\left(\mathbf{S}_1^{1/2}\mathbf{S}_2\mathbf{S}_1^{1/2}\right)^{1/2}\right)}. \tag{2.5}$$

Note that this distance is also well-defined for positive *semi*definite matrices, contrary to the affine-invariant metric which requires the matrices to be positive definite.

2.2.2 Computing means on the manifold

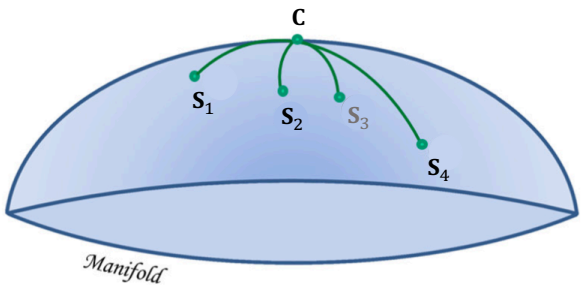


Figure 2.6: Illustration of the geometric mean $\mathbf{C} = \mathcal{G}(\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4)$ *on a Riemannian manifold. Adapted from [Lot+18]*

.

Distance-based classification methods often require to compute a representative (or "center") for a set of SPD matrices $\mathbf{S}_1, \ldots, \mathbf{S}_N$. A simple approach is to take the standard *arithmetic mean*

$$\mathcal{A}(\mathbf{S}_1, \ldots, \mathbf{S}_N) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{S}_i.$$

For the same reason that it does not make sense to measure distances between SPD matrices in the Euclidean space, the arithmetic mean is often not an appropriate center for a set of SPD matrices. Instead, a better approach is to use the more general *geometric mean*

$$\mathcal{G}(\mathbf{S}_1, \dots, \mathbf{S}_N) = \arg\min_{\mathbf{X} \in \mathcal{S}_N^{++}} \sum_{i=1}^{N} \delta^2(\mathbf{S}_i, \mathbf{X}), \tag{2.6}$$

depicted in Fig. 2.6, which allows to choose the most appropriate distance $\delta(\cdot, \cdot)$. Note that the arithmetic mean $\mathcal{A}$ corresponds to the geometric mean $\mathcal{G}$ in the Euclidean metric (2.3). If a manifold is embedded with a Riemannian metric, using the geometric mean in the corresponding distance provides a better center matrix. Unfortunately, in general there is no closed-form solution for the geometric mean for Riemannian metrics; instead, iterative methods must be designed.

Karcher mean for the affine-invariant Riemannian metric The geometric mean with the affine-invariant metric (2.4)

$$\arg\min_{\mathbf{X} \in \mathcal{S}_N^{++}} \sum_{i=1}^{N} \left\| \operatorname{logm}\left(\mathbf{S}_i^{-1}\mathbf{X}\right) \right\|_{\mathrm{F}}^2 \tag{2.7}$$

is commonly called the *Karcher mean*. It has been proved by [Moa05] that it is the unique SPD solution $\mathbf{X}$ of the *Karcher equation*

$$\sum_{i=1}^{N} \operatorname{logm}\left(\mathbf{S}_i^{-1}\mathbf{X}\right) = \mathbf{0}. \tag{2.8}$$

Several iterative algorithms have been explored for computing the Karcher mean. In this work, we use the approach of [BI13] which proposes an algorithm in the spirit of the Richardson iteration for linear systems[2]:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} - \theta \mathbf{X}^{(k)} \sum_{i=1}^{N} \operatorname{logm}\left(\mathbf{S}_i^{-1}\mathbf{X}^{(k)}\right), \tag{2.9}$$

where $\theta > 0$ is a suitable parameter. Note that the solution of the Karcher equation (2.8) is indeed a fixed point of the iteration.

2.3 Low-rank SPD matrices and their manifold

In the previous section, we have identified the space of (full-rank) SPD matrices as a Riemannian manifold, and we have defined some metrics on this manifold. We now turn our attention to *fixed-rank* positive *semi*definite matrices, we are the main interest of this work. First we show how an SPD matrix can be approximated by a low-rank matrix, and how these low-rank approximations can be represented comptactly. Then we present the extension of the Bures-Wasserstein metric on the manifold of fixed-rank positive semidefinite matrices, and explain how we compute means in the sense of this metric.

[2]The Richardson iteration seeks the solution of a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ with the iteration $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \theta\left(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}\right)$. It is actually a gradient descent for the least-squares cost function.

2.3.1 Low-rank approximation of symmetric positive definite matrices

Covariance matrices obtained from EEG signals generally have a low-rank structure. This observation suggests to use a representation of these matrices which is adapted to their structure: a *low-rank approximation*. We will see later that this representation, coupled with an appropriate metric, offers advantages not only in terms of computation time and memory usage, but also in classification accuracy. Let $\mathbf{X} \in \mathcal{S}_n^{++}$ be a SPD matrix. Let $\lambda_1 > \cdots > \lambda_n$ be its eigenvalues, all real and strictly positive, and $u_1, \ldots, u_n \in \mathbb{R}^n$ their associated eigenvectors. This matrix admits the eigendecomposition

$$\mathbf{X} = \lambda_1 \Big(u_1 u_1^\top \Big) + \cdots + \lambda_n \Big(u_n u_n^\top \Big). \tag{2.10}$$

The number of terms of this decomposition corresponds to the rank of the matrix: indeed, if $\mathbf{X}$ has rank r then only the first r eigenvalues are non-zero. A matrix obtained from real signals will never be exactly low-rank, but can be "almost low-rank" in the sense that some eigenvalues are negligible compared to the others. We then say that the matrix is *ill-conditioned* since its condition number, λ_1/λ_n, is large.

In this case, it is useful to approximate the matrix $\mathbf{X}$ by a matrix $\widehat{\mathbf{X}}$ of low-rank $r < n$ fixed. A natural choice of approximation is to cancel its $n - r$ smallest eigenvalues:

$$\mathbf{X} \approx \widehat{\mathbf{X}} = \lambda_1 \Big(u_1 u_1^\top \Big) + \cdots + \lambda_r \Big(u_r u_r^\top \Big).$$

It turns out that this approximation is the best in the sense of the Frobenius norm; formally, it is the solution of the optimization problem

$$\min_{\widehat{D} \in \mathbb{R}^{n \times n}} \left\| \mathbf{X} - \widehat{\mathbf{X}} \right\|_{\mathrm{F}} \quad \text{such as} \quad \operatorname{rank}(\widehat{\mathbf{X}}) \leq r.$$

This result is known as the Eckart-Young theorem.

Compact representation By writing the low-rank approximation $\widehat{\mathbf{X}}$ in matrix form, we can find a compact representation of it. In matrix form, the eigendecomposition (2.10) reads

$$\mathbf{X} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top,$$

where $\mathbf{U} = \Big(\ u_1 \ \Big| \ \ldots \ \Big| \ u_n \ \Big)$ and $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. By setting $\lambda_{r+1} = \cdots = \lambda_n = 0$, we find for the low-rank approximation

$$\widehat{\mathbf{X}} = \mathbf{U}_r \mathbf{\Lambda}_r \mathbf{U}_r^\top,$$

where $\mathbf{U}_r = \Big(\ u_1 \ \Big| \ \ldots \ \Big| \ u_r \ \Big)$ and $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_r)$. Going further, we have

$$\widehat{\mathbf{X}} = \Big(\mathbf{U}_r \mathbf{\Lambda}_r^{1/2} \Big) \Big(\mathbf{U}_r \mathbf{\Lambda}_r^{1/2} \Big)^\top \triangleq \mathbf{Y} \mathbf{Y}^\top,$$

with $\mathbf{Y} \in \mathbb{R}_*^{n \times r}$ (the set of full-rank $n \times r$ matrices). Note that this factorization of $\widehat{\mathbf{X}}$ is not unique: if $\mathbf{Q}$ is an $r \times r$ orthogonal matrix, then $(\mathbf{YQ})(\mathbf{YQ})^\top = \mathbf{Y}\mathbf{Y}^\top = \widehat{\mathbf{X}}$.

2.3.2 Metric for the manifold of fixed-rank positive semidefinite matrices

In this work, we focus on the use of low-rank approximations of covariance matrices. In general, we cannot exploit metrics of the SPD manifold on these low-rank matrices since they are not positive definite (and not invertible). Instead, we work with the manifold of $n \times n$ positive semidefinite matrices of fixed rank r, denoted $\mathcal{S}^+_{n,r}$. Recently, in [MA20], a generalization of the Bures-Wasserstein metric was proposed for this manifold: for $\mathbf{S}_1, \mathbf{S}_2 \in \mathcal{S}^+_{n,r}$:

$$\delta(\mathbf{S}_1, \mathbf{S}_2) = \sqrt{\operatorname{tr}(\mathbf{S}_1) + \operatorname{tr}(\mathbf{S}_2) - 2\operatorname{tr}\left((\mathbf{S}_1^{1/2}\mathbf{S}_2\mathbf{S}_1^{1/2})^{1/2}\right)}, \tag{2.11}$$

which is the same expression as the full-rank metric Eq. (2.5). The distance can also conveniently be expressed in terms of compact representations $\mathbf{S}_1 = \mathbf{Y}_1\mathbf{Y}_1^\top$, $\mathbf{S}_2 = \mathbf{Y}_2\mathbf{Y}_2^\top$. We define the singular value decomposition $\mathbf{Y}_1^\top\mathbf{Y}_2 = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ and $\mathbf{Q}^* = \mathbf{V}\mathbf{U}^\top$. Then,

$$\delta(\mathbf{S}_1, \mathbf{S}_2) = \|\mathbf{Y}_2\mathbf{Q}^* - \mathbf{Y}_1\|_{\mathrm{F}}.$$

Note that this distance is invariant under rotations of $\mathbf{Y}_1$ and $\mathbf{Y}_2$: indeed, replacing with $\mathbf{Y}_1\mathbf{Q}_1$ and $\mathbf{Y}_2\mathbf{Q}_2$ yields the same distance (this can be shown with the fact that the Frobenius norm is invariant under rotation). Consequently, the distance function d has the *orthogonal-invariance* property: for $\mathbf{W}$ orthogonal,

$$\delta\Big(\mathbf{W}^\top\mathbf{S}_1\mathbf{W}, \mathbf{W}^\top\mathbf{S}_2\mathbf{W}\Big) = \delta(\mathbf{S}_1, \mathbf{S}_2). \tag{2.12}$$

Orthogonal-invariance is weaker than affine-invariance because orthogonal matrices are invertible, but the converse is not true.

2.3.3 Computing means in the low-rank metric

As the fixed-rank metric that we have introduced is fairly new, algorithms for computing the Riemannian mean of fixed-rank SPD matrices have not yet been explored. We propose a straightforward method relying on the manifold optimization toolbox *Manopt* [Bou+14]. Knowing the expression of the manifold gradient at any point $\mathbf{X}$, the method minimizes the cost function (2.6) with a limited-memory BFGS solver. We do not formally derive the manifold gradient; instead, we give the expression and explain the intuition. Let us decompose the objective function $f(\mathbf{X}) = f_1(\mathbf{X}) + \cdots + f_N(\mathbf{X})$ such that

$$f_i(\mathbf{X}) = \frac{1}{2}\delta^2(\mathbf{S}_i, \mathbf{X}). \tag{2.13}$$

Then, for every i, the steepest descent direction $-\boldsymbol{\nabla} f_i$ is the tangent vector at $\mathbf{X}$ pointing towards $\mathbf{S}_i$, obtained by taking the logarithmic map of $\mathbf{S}_i$ onto the tangent plane at $\mathbf{X}$:

$$-\boldsymbol{\nabla} f_i(\mathbf{X}) = \operatorname{Log}_{\mathbf{X}} \mathbf{S}_i. \tag{2.14}$$

By adding up the gradients of each f_i we obtain the complete gradient expression. The logarithmic map Log and the L-BFGS solver are provided by Manopt.

Chapter 3

Brain-Computer Interfaces and Classification of EEG Signals

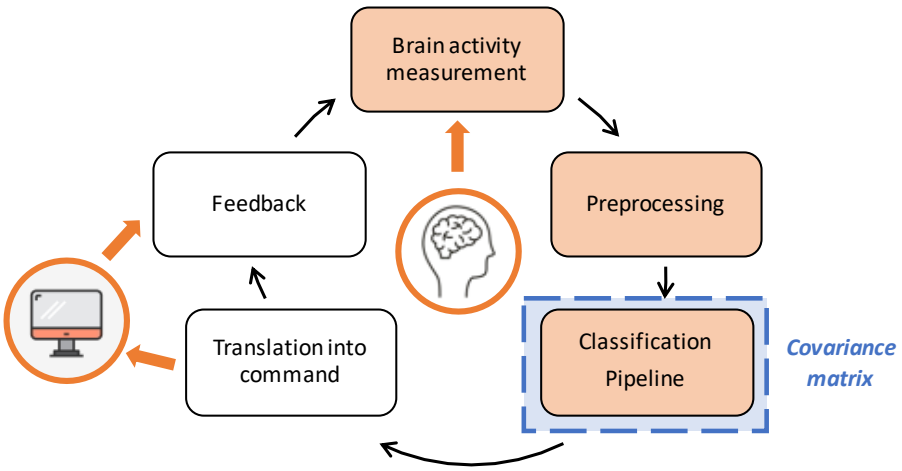


Figure 3.1: Typical feedback loop for a Brain-Computer Interface.

As the name suggests, Brain-Computer Interfaces (BCI) enable their users to interact with computers via brain activity only. A typical BCI consists of several steps and one fundamental concept, the covariance matrix, that form a feedback loop between the user and the computer; see Fig. 3.1 [LBC15]. The main steps will be quickly introduced below but for the rest of the chapter we will focus on the following three steps, represented in orange in the figure: brain activity measurement, preprocessing and classification pipeline and the concept of covariance matrix.

Brain activity measurement In order to communicate with computers, BCIs take signals generated by the brain as input. This work focuses on EEG signals, the most popular input choice in BCI.

Preprocessing The input signals may contain some artifacts (i.e. noise), it is therefore necessary to ensure that only relevant information from the signals is present.

Covariance matrix One of the fundamental concepts for the pipeline classification step is the covariance matrix. It will allow to represent the signals.

Classification pipeline The term "classification pipeline" refers to all the steps required to classify a new signal, i.e. assign it to a *class*. Each brain activity pattern defines a class. *Features* are characteristics of the signals that allow to discriminate between the different classes. First, appropriate features are extracted from the signal and, using a classifier, a class is assigned based on the features.

Translation into command Each class is associated with a command (e.g. move the cursor to the right if the signal is in the right hand movement class) that can be used to control several devices or applications.

Feedback After an action has been performed, it informs the user about the perception of the signals he has sent and helps to improve the control of the BCI.

The goal of this chapter is to give an overview of how common brain-computer interfaces work in general: we start with a brief introduction to brain activity measurement and shortly explain how the preprocessing of signals is done. After that we review the most common methods for estimating covariance matrices of signals. Finally, we dive into the classification pipeline which is the most relevant for this work. This classification pipeline can be composed of two to three steps depending on the approach used. Two main approaches exist, the classical approaches and the Riemannian approaches. We present the state-of-the-art approaches to both classical and Riemannian EEG classification.

3.1 Brain activity measurement

In order to understand the signals that BCIs process, we must briefly introduce the required neuroscience concepts. We quickly review the anatomy of the brain, then we see that it is possible to measure brain activity in different ways, the most popular of which is electroencephalography (EEG) which we describe in more detail. Finally, we see 3 patterns of brain activity that we can use to generate these EEG signals.

3.1.1 Anatomy of the brain

The brain is composed of 3 main parts: the *cerebrum* which is the largest part of the brain and includes the cortex; the *cerebellum*, and the *brainstem*.

The cortex is divided into left and right hemispheres, each with 4 lobes: frontal, parietal, occipital and temporal, as we can see in Figure 3.2. The frontal and parietal lobe ares separated by the central sulcus.

Each lobe is associated with some body functions located in specific areas of the lobe. The frontal lobe is associated with personality traits but also concentration, planning, problem solving, motor control, speech and smell. The parietal lobe is used to identify objects, body awareness and taste. It can also perceive tactile simulations such as touch and pressure. The memory but also hearing, reading and facial recognition are performed by the temporal lobe. Finally, the occipital lobe focuses on vision.

The knowledge of the location of different functions in the brain will prove to be very useful for the rest of this work. Indeed, in the majority of EEG-based BCI, only the electrodes located near the brain regions active during an experiment will be considered.

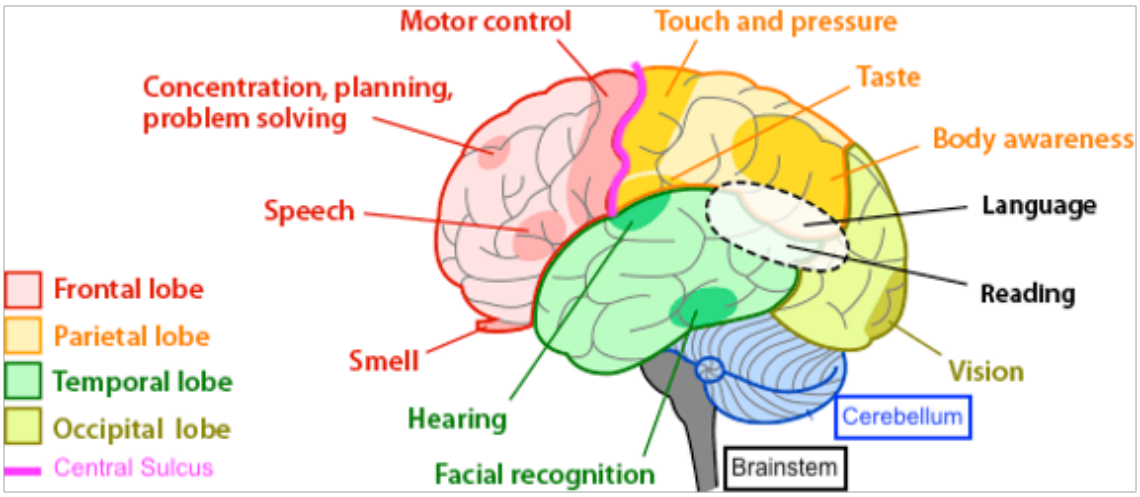


Figure 3.2: Left hemisphere of the cortex, the four lobes that compose it and the associated body functions. Adapted from [Szy11]

.

3.1.2 Brain activity measures classification

Dependability	***Operating mode***	***Recording technique***
• Dependent • Independent	• Synchronous • Asynchronous	• Invasive • Non-invasive

Figure 3.3: Classification of brain signals measures following three criteria.

In order to communicate with computers, BCIs take signals generated by the brain as input. There are many systems that we can use to acquire these signals. These systems can be classified according to different criteria such as dependability, operating mode and recording technique, as represented Fig. 3.3 [Ras+20].

Dependability A BCI is said to be *dependent* if a muscular activity of the user is necessary to generate the signals. An example of dependent BCI is one where a user is required to look at a position on the screen where he wants to move a cursor. Contrarily, if the user does not move his eyes but just needs to imagine hand movements to move the cursor, then the BCI is *independent*. Independent BCIs are more challenging to design but are ideal for severely impaired patients.

Operating mode The mode of operation of the BCI defines its temporal dependency on the user. If the user can only interact with the BCI for a specific period of time, it is a *synchronous* BCI. On the other hand, if the user can interact with the system whenever he wants, it is not time-dependent and it is therefore an *asynchronous* BCI. Synchronous BCIs exist mostly in laboratories, for example when a brain signal is labeled with the

user's intended action. A user controlling a wheelchair uses an asynchronous BCI because the system can be interacted with at any time. Asynchronous BCIs are more practical for users, but challenging to design since the system needs to automatically discriminate when the user is interacting.

Recording technique The signals can be recorded *invasively* or *non-invasively*. In the first case, a surgical operation is necessary to place the sensors inside the skull while in the other case the signals are measured directly on the scalp and therefore no surgical manipulation is required. The most common invasive techniques in BCI are intracortical recording and electrocorticography (ECoG). Despite the quality of the signals generated, the complications due to the installation and maintenance of these techniques mean that they are mostly used on patients for clinical diagnosis or animals. There exists many non-invasive techniques among which we can mention five based on three different concepts. First, the functional magnetic resonance imaging (fMRI) and functional near infrared spectroscopy (fNIRS) which take into account the blood oxygenation. Then, based on nuclear medicine, the positron emission tomography (PET). And finally, by measuring electromagnetic fields, magnetoencephalography (MEG) and electroencephalography (EEG).

3.1.3 Electroencephalography

Electroencephalography (EEG) is a non-invasive functional brain imaging technique that measures the electrical activity generated by nerve cells through electrodes placed on the scalp. The electroencephalogram is the transcription in the form of a trace of the variations over time of the electrical activity of the brain. EEG can be used to study brain function in healthy individuals, but also to diagnose certain diseases that alter brain electrical activity (e.g. epilepsy, migraines, sleep disorders,...).

This brain activity can be captured using electrodes placed on the scalp by means of a helmet/cap (Figure 3.4) and in some cases a conductive paste used to reduce the impedance i.e. the resistance to the flow of an electric current.

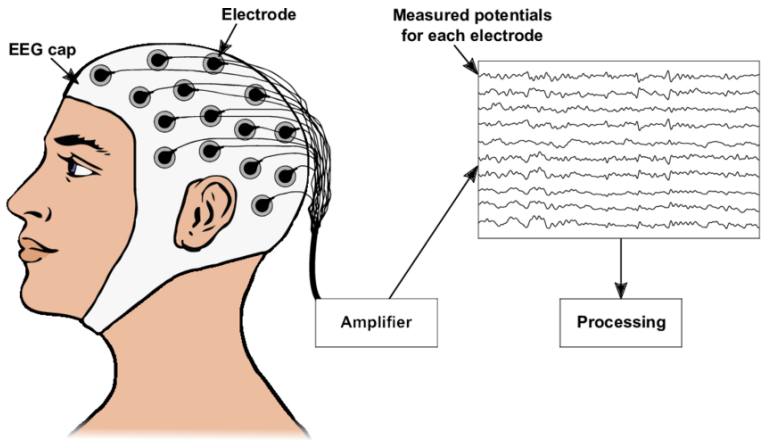


Figure 3.4: Electroencephalography procedure. From [Nag19].

Active vs. Passive electrodes There exists 2 types of electrodes. *Active electrodes* have a pre-amplification system inside the electrode itself and do not require any preparation of the skin (e.g. conductive paste), unlike *passive electrodes* which have no amplification system and require no preparation of the skin.

Unipolar vs. bipolar mounting The term mounting designates a combination of pairs of electrodes.

The distinction between the two mountings comes from the way the signals are obtained from the electrodes: in a *unipolar assembly*, there is a unique reference electrode and a signal is obtained as the potential difference between an active electrode and the reference electrode. In a *bipolar assembly*, a signal is obtained as the potential difference between pairs of electrodes, without a unique common reference. Other mounting schemes exist, for example where the average of all the measured signals is used as reference electrode [BS20].

Electrode placement The international 10/20 system of electrode placement is the most commonly used system. The name of the placement comes from the fact that the distance between two adjacent electrodes is 10% to 20% of the total distance, either between the front and back of the head or between the left and right sides of the skull. There exists also other systems for placing electrodes, such as the 10/10 system.

Amplification Since EEG signals have a low amplitude, in the microvolt range, they must be amplified using an amplification device to obtain measurable signals. For this purpose, a selective amplification device with a high gain is used. The gain is the ratio of the output signal to the input signal (expressed in dB). The amplifiers also remove noise in the EEG signal that affects the electrodes.

Filtering EEG signals can very easily be distorted by noise. In order to reduce this negative impact, different filters can be used such as a high-pass filter to attenuate low frequencies or a low-pass filter to eliminate rapid high frequency variations.

3.1.4 Brain activity patterns in EEG-based BCIs

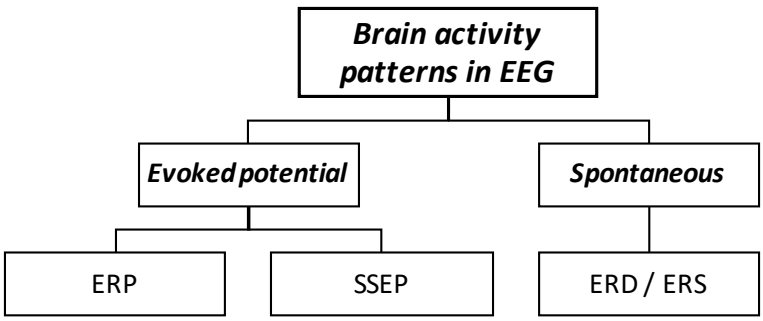


Figure 3.5: Brain activity patterns in EEG.

The use of EEG signals as input signals for BCI is the most popular choice. In addition to having excellent temporal resolution, EEG signals are safe and affordable because they can be acquired using non-invasive sensors. Finally, the device is very easy to set up, which makes it very portable and thus allows for mobile use, outside of laboratory environments.

Three main brain activity patterns are used to design EEG-based BCIs (Fig. 3.5):

Event-related (de)synchronization (ERD/ERS) *Desynchronization* corresponds to billions of neurons firing in a way that is desynchronized with respect to a rest condition; on the contrary, *synchronization* corresponds to neurons firing synchronously. This "firing" is measured via amplitude changes in the power of EEG signals in certain frequency bands. ERD/ERS is typically observed when executing or imagining movements.

Event-related potentials (ERP) A brain activity response is said to be *stereotypical* if it is produced without any particular training of the user (like a reflex). ERPs are stereotypical deflections of the EEG signals (increase or decrease in amplitude) when the user is confronted to a sensory, cognitive or motor event triggered at a precise measurable time. Contrarily to ERD/ERS, event-related potentials are not sustained but only exist during a short time span after the trigger.

Steady-state evoked potentials (SSEP) Another way to produce brain activity is to let the user pay attention to a *periodic stimulus*, such as a flickering light or an amplitude-modulated sound. A steady-state evoked potential corresponds to an increase in EEG power at the frequency of the stimulus. SSEPs are modulated by attention: the more attention the user pays to the stimulus, the higher the SSEP response. When a user is presented multiple stimuli, this technique allows to determine to which stimulus the user is paying the most attention; each stimulus can then be associated to a command.

The most popular SSEPs for BCI design are steady-state visual evoked potentials (SSVEP), which are observed in the visual area of the brain, in the occipital lobe. They require very little user training and can provide the user with large number of commands. The main drawbacks of SSVEPs is that they require a device that produces the visual stimuli, and they rely on the user's visual attention.

3.2 Signal preprocessing

Raw EEG signals are obtained as a result of the brain activity measurement step. Those raw signals may contain information that are not related to brain activity but to other activities instead such as motor activity, eye movement, etc. Even if such information is not harmful and can on the contrary give an additional indication such as discomfort or distraction of the patient, BCIs focus on processing only signals generated by brain activity.

The preprocessing step will be used to clean up the signals and keep only those parts that contain only brain-generated information. It is therefore necessary to first, detect that the signal is carrying information unrelated to brain activity and then process the signal accordingly.

These artifacts can be detected in different ways depending on their origin; if the artifact is muscular, an amplitude threshold can be applied either on the band power or on the time-courses of certain channels. One can also try to capture these artifacts by placing additional electrodes near the presumed locations where they are generated. For example, if an artifact is related to eye vision, one can place electrodes near the patient's eyes to try to detect if activity is generated at that location.

After detecting these artifacts, we must process them. There are two possibilities; either we have to reject them, i.e. remove a part of the signal (or even the whole signal) because the information it contains is too corrupted, or it is not the case and we can therefore try to reduce the effects of these artifacts with adapted techniques or filters. There are two cases where we have to reject a part of the signal: first, if these artifacts are too important compared to the initial signal and we will never be able to compensate their effects with a technique or a filter. Secondly, if these artifacts indicate that the recording conditions for this signal are not correct, then that part of the signal is not usable either.

3.3 Covariance matrix estimation

The signal spatial covariance is at the heart of most approaches for EEG classification. The covariance matrix is unknown and must be inferred from the EEG samples. Estimating the covariance is not a trivial task: the choice of the appropriate estimator is crucial in order for the covariance matrices to fulfill the required properties: they should be

- *accurate* (close to the true covariance),
- *symmetric* and *positive definite*; in particular, all eigenvalues must be positive,
- *well-conditioned* (the ratio between the maximum and minimum eigenvalue must not be too large.

Let us begin by formally defining the covariance of EEG signals. A multichannel EEG trial recorded on C electrodes consists of N samples $\mathbf{x}_n \in \mathbb{R}^C$, $n = 1, \ldots, N$. The assumption is made that all N samples $\mathbf{x}_n$ are realizations of a random vector $\mathbf{x}$. Let $\bar{\mathbf{x}} = \mathrm{E}[\mathbf{x}]$ denote the expected vector. The covariance matrix

$$\boldsymbol{\Sigma} = \mathrm{E}\Big[(\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^\top\Big]$$

gives the covariance between each pair of elements of the random vector. This matrix is *symmetric* and *positive definite* (provided that $\mathbf{x}$ is not a constant random vector, in which case the covariance is the zero matrix). Indeed,

$$\Sigma_{ij} = \mathrm{E}[(x_i - \bar{x}_i)(x_j - \bar{x}_j)] = \mathrm{E}[(x_j - \bar{x}_j)(x_i - \bar{x}_i)] = \Sigma_{ji}$$

and for any $\mathbf{y} \in \mathbb{R}^C \setminus \{\mathbf{0}\}$,

$$\mathbf{y}^\top \boldsymbol{\Sigma} \mathbf{y} = \mathrm{E}\Big[\mathbf{y}^\top (\mathbf{x} - \bar{\mathbf{x}})(\mathbf{x} - \bar{\mathbf{x}})^\top \mathbf{y}\Big] = \mathrm{E}\Big[\Big(\mathbf{y}^\top (\mathbf{x} - \bar{\mathbf{x}})\Big)^2\Big] > 0.$$

Several estimators have been developed for computing the covariance given a set of samples $\mathbf{x}_n$. We start by defining the *sample covariance matrix* which is the most usual estimator. We then present the class of *shrunk covariance* estimators which are shown to yield better properties than the former, and are particularly useful in the context of BCIs.

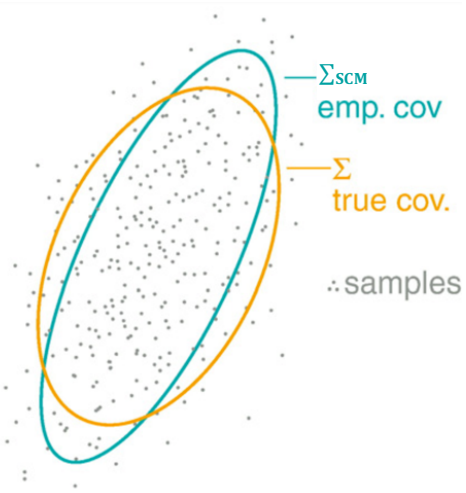


Figure 3.6: Data points drawn from a 2d Gaussian distribution with the covariance represented by the orange ellipsoid, and estimated sample covariance in blue. The sample covariance is elongated; shrinkage alleviates this effect by shrinking the covariance towards a circle. Adapted from [Bla+11].

Sample covariance matrix estimator The most usual estimator is the empirical sample covariance matrix (SCM) defined as

$$\mathbf{\Sigma}_{\text{scm}} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top, \tag{3.1}$$

where $\bar{\mathbf{x}} \in \mathbb{R}^C$ is the sample mean vector $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$ (note that we abuse notations with the true expected vector $\mathrm{E}[\mathbf{x}]$). The sample covariance is computationally simple and *consistent*, which means that as the number of observations N grows, the estimator converges towards the true covariance $\mathbf{\Sigma}$. However, it is known to perform poorly when the dimension C of the matrix is large. When $C > N$ the matrix is rank-deficient: indeed, it is the sum of N rank-one matrices $(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$, hence the rank of $\mathbf{\Sigma}_{\text{scm}}$ is at most N. Even when $C \leq N$ the estimator is often ill-conditioned if C is not much smaller than N. Therefore the sample covariance is not the best one to work with when dealing with large-dimensional covariance matrices.

Shrunk covariance matrix estimators The shrunk covariance estimator was initially proposed by [Ste56] as a way to overcome the ill-conditioning of the sample covariance estimator. He noticed that the sample covariance estimator tends to overestimate the largest eigenvalues and underestimate the smallest ones. To overcome this, he "shrinks" the eigenvalue spectrum towards the mean eigenvalue $\nu = \mathrm{tr}(\mathbf{\Sigma}_{\text{scm}})/C$ by taking a convex combination of the sample covariance and the diagonal matrix $\nu\mathbf{I}$:

$$\mathbf{\Sigma}_{\text{shrink}} = (1-\rho)\mathbf{\Sigma}_{\text{scm}} + \rho\nu\mathbf{I}. \tag{3.2}$$

This convex combination actually performs a convex combination of each eigenvalue with the average eigenvalue ν: with the eigendecomposition $\mathbf{\Sigma}_{\text{scm}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$,

$$\mathbf{\Sigma}_{\text{shrink}} = (1-\rho)\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top + \rho\nu\mathbf{I} = (1-\rho)\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top + \rho\nu\mathbf{U}\mathbf{I}\mathbf{U}^\top = \mathbf{U}((1-\rho)\mathbf{\Lambda} + \rho\nu\mathbf{I})\mathbf{U}^\top. \tag{3.3}$$

One still needs to determine which is the best shrinkage parameter ρ to use. Analytic methods for computing optimal shrinkage parameters have recently emerged. One of the best-known was proposed by Ledoit and Wolf [LW04] which formulate this task as an optimization problem:

$$\min_{\rho,\nu} \mathrm{E}\Big[\|\boldsymbol{\Sigma}_{\text{shrink}} - \boldsymbol{\Sigma}\|^2\Big]. \tag{3.4}$$

Note that this problem is slightly more general since ν is a variable of the optimization. Since $\boldsymbol{\Sigma}$ is unknown a priori, this optimization problem cannot be solved directly. Instead, the authors give estimators for ρ and ν which are *asymptotically consistent*, which means that they converge towards the optimizer of (3.4) when *both* N and C grow together. This notion of asymptotic convergence provides an estimator that performs well even with large-dimensional matrices. The optimal parameters are given by

$$\nu = \mathrm{tr}(\boldsymbol{\Sigma}_{\text{scm}})/C, \qquad \rho = \frac{\frac{1}{N^2}\sum_{n=1}^{N} \Big\|(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top - \boldsymbol{\Sigma}_{\text{scm}}\Big\|^2}{\|\boldsymbol{\Sigma}_{\text{scm}} - \nu\mathbf{I}\|}. \tag{3.5}$$

Other approaches for computing ρ have been proposed in the literature, e.g., [SS05], but in this work we focus on the Ledoit-Wolf estimator as it is the most popular one.

The first application of shrunk covariance estimators in brain-computer interfaces was done by [Vid+09]. It led to the development of the shrinkage LDA (sLDA) classifier which is a standard LDA classifier (explained in Section 3.5) with covariance matrices regularized by shrinkage [Bla+11]. sLDA was found to be effective with little training data, and superior to the standard LDA classifier, making it one of the state-of-the-art EEG classifiers in the non-Riemannian realm [Lot+18].

Since shrunk covariance estimators help to regularize rank-deficient matrices, they are particularly interesting for the present work as they can be integrated in a Riemannian approach to treat high-dimensional covariance matrices. Later, in our experiments, we evaluate how Riemannian approaches benefit from this regularization, and how this improvement compares to directly using the fixed-rank Riemannian metric on low-rank approximations of the covariance matrix.

3.4 Classical vs. Riemannian approaches to EEG classification

The classification pipeline step is the part of the BCI feedback loop that interests us most in this thesis.

Earlier in the text, we considered this step as a black box, we will now analyze it in detail. Its purpose is to classify EEG signals. For this, two approaches exist: classical and Riemannian approaches. Both approaches use the representation of signals as a covariance matrix. However, as we will see, Riemannian approaches use these matrices directly for classification, unlike classical approaches. Moreover, the classical approaches are composed of three phases while the Riemannian approaches have only two as we can see in figure 3.7 [YBL17].

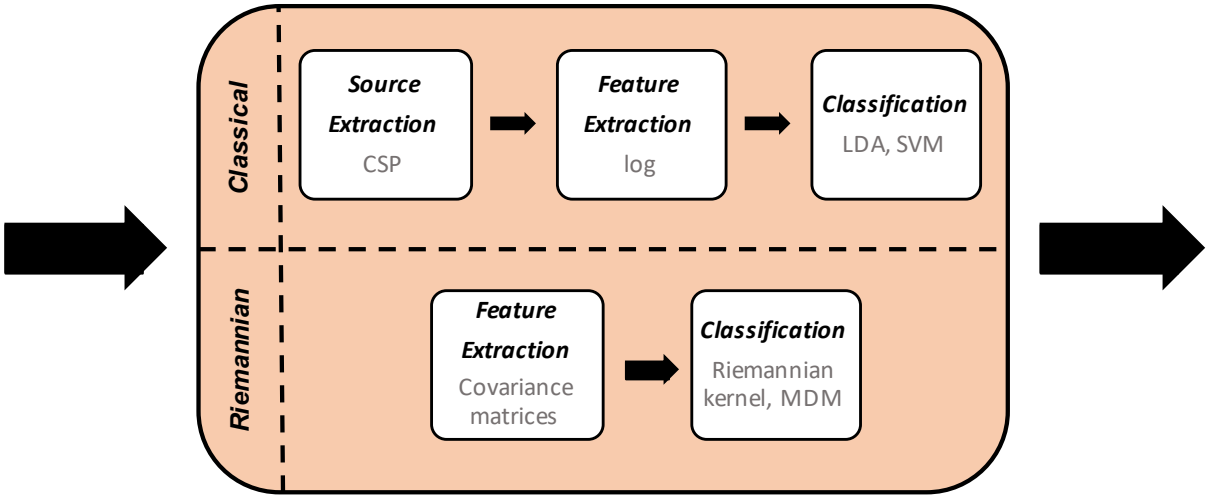


Figure 3.7: Detailed classification pipeline step divided in two parts depending on the approach used: either classical or Riemannian. Grey text is used for methods, concepts examples.

3.5 Classical approaches to EEG classification

In what follows, we will first consider the classical approaches, represented in the top of figure 3.7. These are composed of three steps: sources extraction, feature extraction and the classification.

3.5.1 Source extraction

The final goal of our pipeline will be to classify the EEG signals in order to generate a command. But before we can classify them, we must first find criteria that will allow us to differentiate the different signals: *features*. These features, in addition to being measurable for each signal, must also allow us to highlight the similarities within the same class and conversely to highlight the differences between two distinct classes.

Depending on the signals on which the BCIs are based, specific combination of features are recommended. For BCIs based on *oscillatory activity*, i.e., a change in band frequency power appearing in specific areas of the brain, such as ERD/ERS or SSVEP, features based on *spectral and spatial information* are preferred. On the opposite, for BCIs based on *ERP*, features based on *temporal and spatial information* are recommended. In what follows, we will mainly focus on BCIs based on oscillatory activity as these are the ones we are interested in in this work.

There is a family of features, called *power band*, which allows to take into account the spectral and spatial information of a signal. The simplest way to calculate a power band feature for a particular channel, denoted by $P_{x-\mu}$ with x the chosen channel and μ the filtering frequency, is the following and consists in three steps: filter the EEG signal from channel x according to a chosen frequency μ, perform a power estimation of the signal by taking its square and finally, perform a temporal average.

This method of calculating the power band does not take into account the different channels and is therefore not optimal. Its extension to several channels is possible but

results in a complex optimization calculation. Another way to take into account different channels is to first select relevant features, then do the same with the channels and finally combine channels together and extract features from this new channel, this is called *spatial filtering*.

Let $x_i(t)$ denote the EEG signal of channel i. Spatial filtering consists in computing a weighted sum of the channels

$$\tilde{x}(t) = \sum_i w_i x_i(t) = \mathbf{w}^\top \mathbf{x}(t).$$

In the BCI literature this spatial filtering is called *source extraction*. There are different ways to choose the weights w_i but the most used is the *Common Spatial Patterns* (CSP) procedure. Let $\mathbf{X}$ be the $C \times N$ matrix representing the discretized EEG signal with C channels and N time samples. Given two signals $\mathbf{X}_1$, $\mathbf{X}_2$ (corresponding to two classes to discriminate), CSP finds the weights w_i by extremizing the variance difference between the filtered signals:

$$J_{\text{CSP}}(\mathbf{w}) = \frac{\left\|\mathbf{w}^\top \mathbf{X}_1\right\|^2}{\left\|\mathbf{w}^\top \mathbf{X}_2\right\|^2} = \frac{\mathbf{w}^\top \mathbf{\Sigma}_1 \mathbf{w}}{\mathbf{w}^\top \mathbf{\Sigma}_2 \mathbf{w}}, \tag{3.6}$$

where the sample covariance matrix $\mathbf{\Sigma}_k = \frac{1}{N-1}\mathbf{X}_k\mathbf{X}_k^\top$ appears naturally. When multiple signal trials are available for a class, their covariance matrices are simply averaged. For extremizing J_{CSP} the generalized eigendecomposition (GED) of $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$, i.e., the eigenvalue problem $\mathbf{\Sigma}_1\mathbf{w} = \lambda\mathbf{\Sigma}_2\mathbf{w}$, is computed. The spatial filters which maximize (resp. minimize) J_{CSP} are the eigenvectors corresponding to the largest (resp. smallest) eigenvalues.

3.5.2 Feature extraction

A common way to build a feature vector $\mathbf{f}$ is to compute the filters $\mathbf{w}_i$ corresponding to the 3 largest and 3 smallest eigenvalues. Then, the feature vector of a new signal with covariance $\mathbf{\Sigma}$ is computed as

$$f_i = \log\left(\mathbf{w}_i^\top \mathbf{\Sigma} \mathbf{w}_i\right). \tag{3.7}$$

However, the CSP has some limitations: it is very sensitive to external environment (i.e. noise, instabilities and artifacts). In addition, there are risks of overfitting if the size of the training dataset is not large enough.

3.5.3 Classification

Now that we have defined the different features to differentiate signals, they can be classified. The purpose of the classification is the following: to assign to each new signal its corresponding class. To do so, we will proceed in two steps: first, from a training set composed of N data points where each point is a couple of a vector of F features $\mathbf{f}_n \in \mathbb{R}^F$ and its class $c_n \in C$, noted $(\mathbf{f}_n, c_n)$, we will learn a classification function $\mathcal{F} : \mathbb{R}^F \to C$, which allows to assign a class c to a new feature vector $\mathbf{f}$.

Depending on the number of classes in C, the classification is said to be *binary* if $|C| = 2$ or *multi-class* if $|C| > 2$. The binary classification is the most straightforward of the two because direct binary techniques can be used. Regarding multi-class classification,

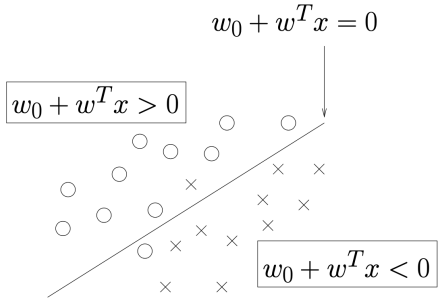


Figure 3.8: Two classes of elements are "separated" by a hyperplane. From [Lot+07].

there are two different ways to proceed: either directly use multi-class techniques such as decision trees and get the desired class, or decompose the multi-class problem into several binary problems easier to solve. One such strategy is "One Versus the Rest" (OVR): for each class c, a binary classifier is trained to discriminate class c from the other classes. If the binary classifiers have a confidence score, we simply pick the class that is the most confidently discriminated from the other classes.

A common binary classification method, often used for EEG classification, is to separate the two classes by a hyperplane as depicted in Fig. 3.8. The sign of the hyperplane equation allows to easily compute the distance from a data point to the plane. There exists several approaches to find the hyperplane separating the classes. A popular one is *Linear Discriminant Analysis* (LDA). LDA assumes that the data are normally distributed, and tries to project the data points on a one-dimensional space that minimizes the overlap of the two classes after projection. This 1d space then gives the vector normal to the separating hyperplane.

Before Riemannian approaches were employed in EEG classification, the CSP+LDA classifier was the state-of-the-art approach [YBL17].

3.6 Riemannian approaches to EEG classification

Before Riemannian approaches were exploited for BCI applications, most successful approaches had in common that they work with covariance matrices instead of raw input signals. As we have seen, Common Spatial Patterns (CSP) find the spatial filter that maximizes the covariance of one class and minimizes the covariance of the other; Linear Discriminant Analysis is then used to discriminate a new signal. In all cases, the covariance matrices are handled as if they were living in a Euclidean space; for example, CSP requires to compute the Euclidean mean of the training covariance matrices. However, covariance matrices are symmetric and positive definite (SPD) and as we have seen in Chapter 2, they lie on a subset of the Euclidean space with special properties: the SPD manifold. A promising approach is to use a Riemannian metric instead of a Euclidean one. This can be achieved by directly working with covariance matrices as features, and using metric-based classifiers with an appropriate distance function.

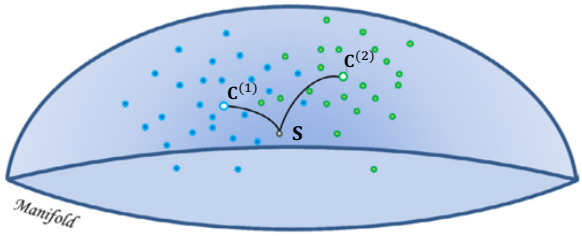


Figure 3.9: Minimum Distance to the Riemannian Mean with two classes. During the training phase, the Riemannian means $\mathbf{C}^{(1)}$ *and* $\mathbf{C}^{(2)}$ *are computed. A new sample* $\mathbf{S}$ *is classified by taking the closest center in the sense of the Riemannian distance. Adapted from [Lot+18].*

3.6.1 Riemannian classifiers

We present two main approaches for building classifiers based on Riemannian geometry.

Minimum distance to the Riemannian mean (MDRM) The simplest approach to design a fully Riemannian classifier was proposed by [Bar+12]. It consists of a Minimum Distance to the Mean (MDM) classifier that uses an appropriate Riemannian distance function. It is illustrated in Fig. 3.9. Concretely, given a labeled training set $\mathbf{S}_i$ of covariance matrices corresponding to one of K classes, and $\mathcal{I}^{(k)}$ the set of indices corresponding to class k, the Riemannian mean (introduced in Chapter 2) of each class k is computed:

$$\mathbf{C}^{(k)} = \mathcal{G}\left(\mathbf{S}_i, i \in \mathcal{I}^{(k)}\right).$$

Once this training phase is done, an unknown sample $\mathbf{S}$ is classified by choosing the class $\widehat{k}$ for which the center $\mathbf{C}^{(\widehat{k})}$ is closest to $\mathbf{S}$, in terms of the Riemannian distance δ_{R}:

$$\widehat{k} = \arg\min_k \delta_{\mathrm{R}}\left(\mathbf{S}, \mathbf{C}^{(k)}\right).$$

This simple yet effective approach was shown to achieve comparable performance to the standard CSP+LDA approach. It is remarkable that a Riemannian approach is able to achieve this without requiring source extraction; indeed, the spatial information is already available in covariance representations, and Riemannian geometry is able to extract this information implicitly.

Riemannian-based kernel The main drawback of the simple MDM method is that it does not take advantage of the more sophisticated nonlinear classification methods which are known to achieve better performance, such as Support Vector Machines (SVM). As a reminder, in the two-class setting, SVMs seek to separate data by finding a hyperplane that is as far as possible from both classes. The hyperplane is defined by the equation

$$h(\mathbf{x}) = b + \langle \mathbf{w}, \mathbf{x} \rangle = 0,$$

where bias b and normal vector $\mathbf{w}$ are learned through an optimization problem. A new sample $\mathbf{x}$ is classified according to the sign of $h(\mathbf{x})$. If the data is not linearly separable, an SVM can be extended to a nonlinear classifier by sending the data points to another high-dimensional transformed space in which they are linearly separable; this is done

through a nonlinear mapping $\phi(\mathbf{x})$. A key insight is that the mapping ϕ need not be known: the only necessary ingredient is a dot product in the transformed space, called *kernel*:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle.$$

The same authors that proposed the MDRM method have presented in [Bar+13] a kernel based on Riemannian geometry that can be readily used in support vector machines. The approach consists in defining a reference SPD matrix $\mathbf{S}_{\text{ref}}$ and to project covariance matrices onto the space tangent to the manifold at $\mathbf{S}_{\text{ref}}$ through the logarithmic map:

$$\phi(\mathbf{S}) = \text{Log}_{\mathbf{S}_{\text{ref}}}(\mathbf{S}).$$

One can then use as kernel the Riemannian metric in the tangent space; in this case the Affine-Invariant Riemannian Metric:

$$k_{\text{R}}(\mathbf{S}_i, \mathbf{S}_j; \mathbf{S}_{\text{ref}}) = \langle \phi(\mathbf{S}_i), \phi(\mathbf{S}_j) \rangle_{\mathbf{S}_{\text{ref}}} = \text{tr}\left(\text{Log}_{\mathbf{S}_{\text{ref}}}(\mathbf{S}_i) \mathbf{S}_{\text{ref}}^{-1} \text{Log}_{\mathbf{S}_{\text{ref}}}(\mathbf{S}_j) \mathbf{S}_{\text{ref}}^{-1} \right).$$

A question that remains is how to choose the reference SPD matrix $\mathbf{S}_{\text{ref}}$. A natural choice is to use the geometric Riemannian mean of the complete set of covariance matrices. It was demonstrated by [TPM08] that the geometric mean is the point where the mapping ϕ on the tangent space leads to the better local approximation on the manifold.

The SVM classifier coupled with a Riemannian-based kernel was shown to significantly outperform (about 6 %) the standard CSP+LDA method.

3.6.2 Limitations of Riemannian approaches in high dimensionality

Even if Riemannian approaches to EEG classification offer better performances and conceptual simplicity compared to previous approaches, they suffer from a major limitation: their performance decreases significantly when the size of the covariance matrices, i.e., the number of EEG channels, increases. This was empirically shown in [YLS15]; they set up a CSP + SVM pipeline where the average covariance matrix for CSP is computed using different metrics, and they compared the classification accuracy on datasets of varying dimensionality. For the highest-dimensional dataset (118 electrodes) they even show that the arithmetic mean outperforms the Karcher mean by almost 6 % in accuracy. As they outlined, the most plausible explanation is that redundancy increases with more electrodes, and the covariance matrices become rank-deficient. Hence the limit of the SPD assumption for the Riemannian metric is reached.

3.6.3 Use of fixed-rank Riemannian metric in BCI

In the recent work [MA20], a Riemannian metric on the manifold of fixed-rank positive semidefinite matrices was proposed. This metric was used in [Sab+19] to perform regression on MEG and EEG brain signals. In particular, the paper compares two Riemmanian approaches:

Fixed-rank Wasserstein metric. The covariance matrices $\mathbf{C}_i$ belong to the manifold $\mathcal{S}_{n,r}^{+}$ of $n \times n$ positive semidefinite matrices of rank $r < n$. The Riemannian mean $\overline{\mathbf{C}}$

w.r.t. the fixed-rank metric is computed, and the covariances matrices are projected onto the tangent space at $\overline{\mathbf{C}}$ using the logarithmic map. Ridge regression is then used to predict the target on the vectorized covariance matrices.

Affine-invariant metric. As a comparison, a more classical AIRM-based approach is proposed. The affine-invariant metric cannot be used directly on the covariance matrices since they are rank-deficient. Instead, they are projected onto the manifold $\mathcal{S}_r^{++}$ of full-rank $r \times r$ SPD matrices using principal component analysis. Then the full-rank matrices are projected on the tangent space at the Riemannian mean, and the target is predicted using ridge regression.

Their experiments show that the two approaches achieve comparable performance. in terms of mean absolute error of regression. This result already indicates that the fixed-rank Wasserstein metric is a viable approach to build Riemannian classifiers for EEG signals.

In this work, we further extend the observations of [Sab+19] by exploring the classification task instead of the regression task. We perform a detailed analysis not only of accuracy but also of time and memory usage. Finally, we also show how the different metrics are impacted by the choice of covariance estimator.

Chapter 4

Experimental Methodology

In this chapter, we will focus on the classification part. First we will introduce two classification methods used in this work. These classification methods are based on the notion of distance. Their particularity is that they can be used by both approaches, classical and Riemannian, provided that the corresponding metrics and distances are used. In order to compare these classification methods, we need to define objective criteria. We will therefore use performance measures based on three different aspects: classification accuracy, computation time and memory space used. Finally we will detail different ways to compute the classification accuracy based on cross-validation.

4.1 Classification

Now that we have expressed the signals as covariance matrices, it is time to classify them. We decided to focus on two distance-based classification methods: the minimum distance to the mean (MDM) and the k nearest neighbours (k-NN).

The mechanism of classification is simple: we train a model over a labeled dataset then we use this trained model to classify new samples.

4.1.1 Minimum distance to the mean (MDM)

The basic idea underlying the first method is the following: we want to define a kind of distance between a sample and one class of samples as the distance between this sample and the mean of the class.

To do so, we proceed as follows:

- Labeled dataset: For every class c, we have the dataset $\{X_i^{(c)}\}_{i=1,...,n_c}$ with n_c the number of samples in class c.
- Training: The goal of this step is to summarize each class in one point. To do so, we compute the mean of all the training points belonging to a class. Those means are called the centers of each class.

  Compute the mean of every class: $\bar{X}^{(c)} = \text{mean}\{X_1^{(c)}, ..., X_{n_c}^{(c)}\}$.

- Classification: Given a test point, we want to predict its class. To do so, we compute the distances between this test point and all the centers computed earlier. The minimum distance will define the predicted class.

  Predicted class of sample X:

$$\arg\min_{c} \text{dist}(X, \bar{X}^{(c)})$$

4.1.2 k-nearest neighbours (k-NN)

The second method is slightly different; it does not use the notion of mean. The classification will be divided in several steps. As before, we want to predict the class of a sample. To do so, we will:

- Compute the distances between this sample and all the training samples.
- Find the k nearest neighbors by sorting those distances and keep the k minimum ones. k is a hyperparameter that we need to fix.
- Find the predicted class by taking a majority vote between the k nearest neighbors. The majority vote will count how many times each class is represented. If there is a tie, the first class encountered will be chosen.

4.2 Performance measures

As a reminder, our goal is to correctly predict the class of a given EEG signal.Since we have several methods and metrics to do that, it is important that we can compare them with each other.

We decided to compare them at 3 different levels: the accuracy of their classification, the computation time and the memory space used. In this section, we will define, for each of the 3 performance criteria, measures/metrics that will allow us to quantify them.

4.2.1 Classification accuracy

The first and most intuitive approach to take into account the classification's accuracy is to compare the predicted class with the real class and count the number of times the prediction is correct. This allows us to obtain the proportion of samples that were correctly classified, in other words, the accuracy of the classifier.

However, this measure is not sufficient because it does not take into account the importance of certain classes compared to others. Let us imagine that we are trying to detect a cancer from medical images. If the classifier wrongly detects a cancer, it is not very serious; however if it wrongly does NOT detect a cancer the consequences could be dramatic. It is therefore important to take this aspect into account.

One way to do so is to use the confusion matrix and the resulting performance measures.

We will first introduce the confusion matrix for binary-class classification problems, as explained in [Shm20a], and then extend the notions we have seen in this part for multi-class classification problems, as in [Shm20b].

Binary-class classification problems

As the name suggests, binary-class classification problems consist in classifying a sample only between two classes often referred to as "Positive (P)" and "Negative (N)". The confusion matrix is therefore very simple; it has two rows, representing the predicted classes and 2 columns representing the actual classes. When the predicted class corresponds to the actual class, the classification is "True (T)", if it is not the case, the classification is "False (F)". So we have four possibilities: TP, TN, FP and FN as showed figure 4.1.

		Actual	
		Positive	Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

Figure 4.1: Confusion matrix for binary-class classification problems.

Now that we have defined our confusion matrix we can notice that we can express the accuracy, introduced before, from this matrix as follow:

- Accuracy: proportion of sample, Positive and Negative, that was correctly classified.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

We can also derive two new performance metrics:

- Precision: proportion of predicted Positives that is truly Positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall: proportion of actual Positives that is correctly classified.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Depending on the application, we may want high precision or high recall. In the example given above about cancer detection, what is important is that as many people as possible with cancer are detected. Then the classifier needs to have a high recall. Now, imagine that our classifier has the task of recommending movies to watch to a user where Positive stands for relevant movies and Negative for irrelevant movies. In this case, we want that most of the recommended videos are relevant to the user, so we want high precision. Generally, you can not have high precision and high recall at the same time, there exists a compromise between the two.

The natural problem that may arise is: when we have to choose between a classifier that has higher precision and one that has higher recall, which one should we choose? That is exactly why F1-score has been defined. It allows us to take into account the precision and the recall and represent them as a single metric which synthesizes them both: the precision and recall's harmonic mean.

$$\begin{aligned}\text{F1-score} &= \texttt{H}(\text{Precision}, \text{Recall}) \\ &= \frac{1}{\frac{1}{2}\left(\frac{1}{Recall} + \frac{1}{Precision}\right)} \\ &= 2\left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}\right)\end{aligned}$$

Multi-class classification problems

The concepts and performances measures seen for binary-class classification can easily be extended to multi-class classification. The confusion matrix will become more general, as we can see figure 4.2, with C the number of classes.

		Actual			
		1	2	...	C
Predicted	1				
	2				
	...				
	C				

Figure 4.2: Confusion matrix for multi-class classification problems.

For each class, TP, TN, FP and FN are computed using algorithm 1 below. This algorithm takes as input **mat**, a $(C \times C)$ confusion matrix, and computes the TP, TN, FP, FN for each class c. As output, the algorithm returns **TP, TN, FP, FN**, $(1 \times C)$ vectors containing the TP, TN, FP, FN for each class.

Algorithm 1 TP, TN, FP, FN computation for a multi-class classification problem

Require: mat
Ensure: TP, TN, FP, FN
  for $i = 1 \rightarrow C$ **do**
    $TP(i) \leftarrow mat(i, i)$
    $TN(i) \leftarrow sum(mat(i+1, i+1))$
    $FP(i) \leftarrow sum(mat(i, :)) - mat(i, i)$
    $FN(i) \leftarrow sum(mat(:, i)) - mat(i, i)$

Now we can easily compute the precision, the recall and the F1-score for each class:

$$\text{Precision(i)} = \frac{\text{TP(i)}}{\text{TP(i)} + \text{FP(i)}}$$
$$\text{Recall(i)} = \frac{\text{TP(i)}}{\text{TP(i)} + \text{FN(i)}}$$
$$\text{F1-score(i)} = 2\left(\frac{\text{Precision(i)} * \text{Recall(i)}}{\text{Precision(i)} + \text{Recall(i)}}\right)$$

At the end, we want to obtain one F1-score for the whole classifier and not for each class. So we need to find a way to express those F1-scores as one single number. There exist different ways to combine them and we will focus on two of them called respectively: the macro-averaged F1-score, macro-F1 for short and the micro-averaged F1-score, micro-F1 for short.

macro-F1 The most intuitive way is to compute the arithmetic mean of all the F1-scores, it is the macro-averaged F1-score:

$$\text{macro-F1} = \text{mean(F1-scores)}$$

The macro-F1 considers each class equally. This is particularly useful when the classes are not well distributed, i.e. when there are many more samples in one class than in another and conversely.

micro-F1 The second way is to take into account all the samples at the same time and that is why we call it "micro-average". First, we need to compute the micro-average precision and micro-average recall and then the micro-averaged F1-score:

$$\text{micro-precision} = \frac{\text{sum(TP(i))}}{\text{sum(TP(i))} + \text{sum(FP(i))}}$$
$$\text{micro-recall} = \frac{\text{sum(TP(i))}}{\text{sum(TP(i))} + \text{sum(FN(i))}}$$
$$\text{micro-F1} = 2\left(\frac{\text{micro-precision} * \text{micro-recall}}{\text{micro-precision} + \text{micro-recall}}\right)$$

We can notice that the FP sum corresponds to all elements of the matrix except the diagonal elements and that the FN also corresponds to all elements of the matrix except the diagonal elements. The micro-precision and the micro-recall are therefore equal and we can simplify the micro-F1 expression:

$$\begin{aligned}\text{micro-F1} &= 2\left(\frac{\text{micro-precision} * \text{micro-recall}}{\text{micro-precision} + \text{micro-recall}}\right) \\ &= 2\left(\frac{\text{micro-precision}^2}{2(\text{micro-precision})}\right) \\ &= \text{micro-precision} = \text{micro-recall}\end{aligned}$$

The micro-precision, the micro-recall and the micro-F1 are all the same and simply correspond to the accuracy of the classifier.

4.2.2 Computation time

Another important aspect to take into account is the computation time. Indeed, when we work with small datasets we often tend to neglect this aspect as it does not play a crucial role but it takes all its sense when we have to work with much larger datasets which is often the case in BCI applications.

It is very important to define this notion of computation time because it is possible to measure a lot of different things. That is why it is necessary to think carefully about what you want to put forward with this measure of time.

Our case, we have a subject and we want to create for him a wheelchair with which he can communicate only with his brain. In order to do this, we will first have to perform experiments on him to obtain experimental data. Then, from this data, we will have to train our classifier. Finally, when it is parameterized, we will have to use it so that the patient can control his wheelchair in real time.

We can therefore notice that we can consider two interesting measures of time:

1. The training time of the classifier.
2. The classification time of a new signal.

Both measures of time are relevant and the choice of one or the other or both depends on the goal we wish to achieve. However, we can already underline the fact that the measurement of the classification time is very interesting because it allows to measure the reactivity time of the wheelchair which will directly impact the user.

4.2.3 Memory usage

Finally, the last performance measure we will consider is the memory space usage.

We will compare the memory usage of the different methods in a purely theoretical way. As before, we can define the used memory space in different ways depending on:

- The channels' number.
- The covariance matrices' rank.
- The EEG signals' size.
- The classification method used.
- ...

We will focus a little on the last point since it is perhaps less obvious than the others to understand. Depending on the classification method used, we can define the "amount of space" that the classifier takes in memory. For the MDM classifier, as we compute the mean covariance matrix, we only need to store 1 covariance matrix per class. Whereas for the k-NN classifier, we need to store all the covariance matrices because for each new signal, we need to compute the distance between this new covariance matrix and all the classifier's covariance matrices. So we can already notice that there will be a significant difference in memory usage between the two methods.

4.3 Model validation

In section 4.2, we defined several measures to evaluate the performance of a classifier. Now we will use these measures to try to predict how the classifier will behave on new data: this is the objective of the validation.

The general principle behind the validation of a model is the following:

1. Based on a training set we will train and build our model: we will use our knowledge of the actual classes to train our model.

2. Then based on a validation set, we will validate our model: we will predict the classes for this set based on our model and then compare them to the actual classes to estimate the performance of the trained classifier.

Following this general principle, in order to obtain a training set and a validation set, we need to randomly split our dataset into two sets. This first approach is the simplest validation method. It is legitimate to ask how big the training set and the validation set should be. In fact, taking a small training set will give us an average classifier, on the other hand, taking a big training set will give us a poor estimation of performance. There exists a trade-off between the two but it is customary to choose a validation set smaller than the training set, as represented figure 4.3.

However, there are two problems with this method:

- The estimated performance will have a large variance because it depends strongly on the chosen training subset.

- The method will not detect if the model overfits the training set because we only train our model on a single training set.

To overcome this, there should be several independent samples. The solution is cross-validation which forms several different subsets with the same dataset, by crossing the data ("cross"). Cross-validation is very useful because it allows multiple validation sets to be drawn from the same database and thus provides a more robust estimate, with bias and variance, of model validation performance.

We can go even further by setting the size of the subdivisions using K, a parameter chosen a priori. It is the "K-fold validation", a cross-validation with fixed subdivisions of size N/K with N the size of the dataset, represented figure 4.3.

Finally we can introduce the "Leave-one-out": a K-fold where $K = N$ which simply means that we are keeping the whole dataset except one element as the training set and that we try to predict this only element. This approach is better than the ones introduced previously because there is no variance. However, since the experiment must be repeated N times, it is very time consuming and therefore cannot be applied to large datasets.

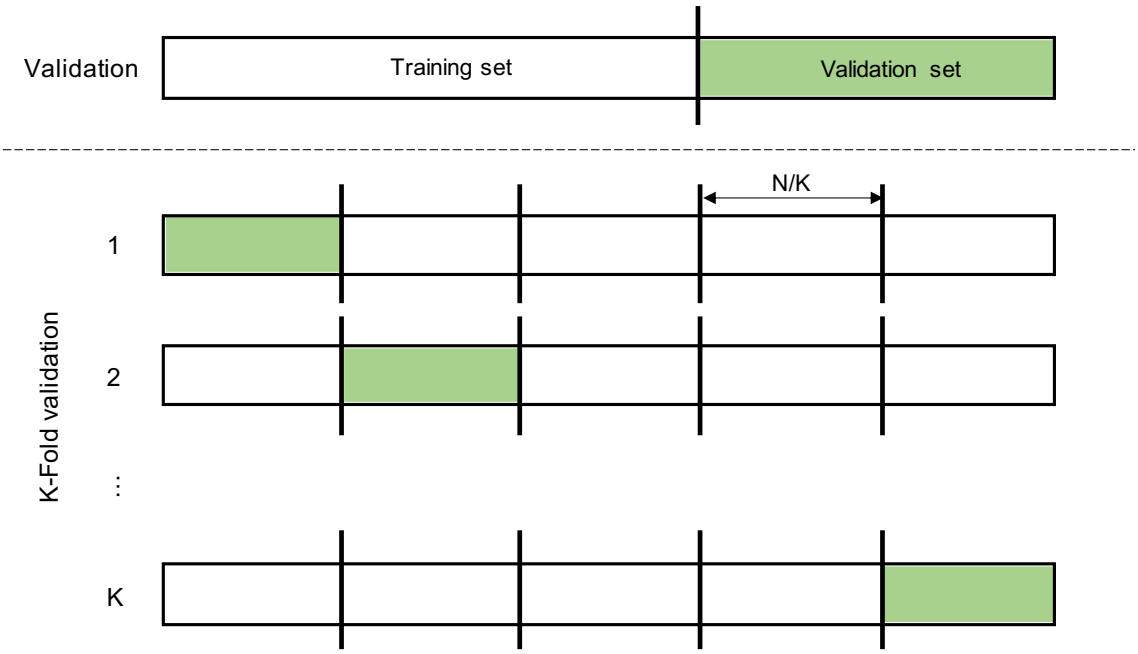


Figure 4.3: Validation and K-fold validation schematic representation.

Chapter 5

Experimental Results

In the previous chapter, we have introduced the tools and the methodology to evaluate classifiers. In this chapter, we conduct several experiments on a real dataset. We start by presenting the dataset and the experimental setup. Then, we show how we optimized the hyperparameter k of the k-Nearest Neighbors classifier. Finally, we focus on two particular aspects that impact the performance: the influence of the covariance matrix estimator and the interest of decreasing the rank of the covariance matrices in the fixed-rank Wasserstein metric.

5.1 Experimental data

We use the dataset available at [Che21] which is part of the MOABB project [JB18]. This dataset contains the EEG of 12 subjects recorded during a SSVEP experiment. For this experiment, 8 electrodes where placed along the scalp of the subject, as represented figure 5.1, and measured the signals with a 256 Hz sampling rate. As we can see, the electrodes are placed at the back of the head. This placement makes sense given the explanations provided in Section 3.1.1 and Section 3.1.4: SSVEP require the subject vision, we can therefore observe them in the occipital lobe.

The experimental device is composed of an electric wheelchair to which a panel is attached on the left. This panel, represented in figure 5.2, is totally visible to the user without him having to move his head. It is composed of three groups of 4 leds and a point, called the fixation point, set at equal distance of the three groups. This point represents the resting class. Each group of leds blinks at its own frequency; respectively $13, 17$ and 21 Hz. We can already notice that there are 4 classes of interest; one for each frequency and one for the resting state.

Once the subject is comfortably seated in the wheelchair, the session can begin. One session is composed of 32 trials: 8 trials for each class. At the beginning of each trial, the subject receives instructions on which group or point to look at. He has to stare at it for 5 seconds then he alternates with a 3 second break and so on until the end of the session. Each subject performs between 2 and 5 sessions. In order to have the same number of trials for each subject, we only consider two sessions per subject, which provides 64 trials per subject.

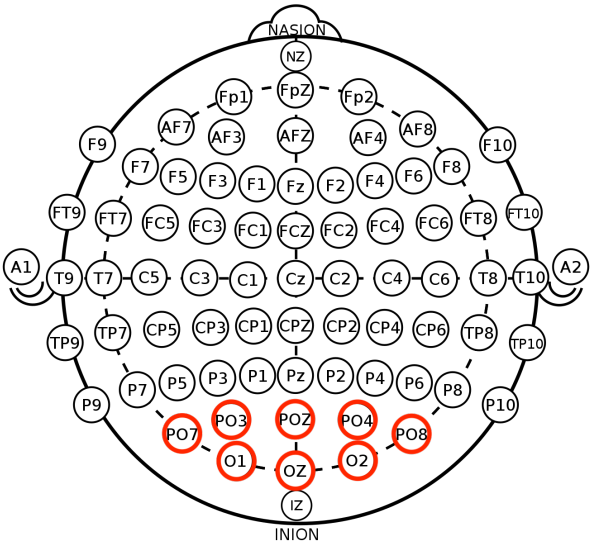


Figure 5.1: Placement of the 8 *electrodes on the subject scalp following the* 10/20 *system.*

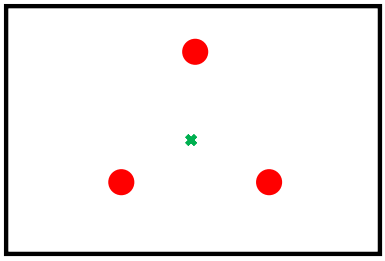


Figure 5.2: Panel of the experimental device. The red dots represent the 3 *groups of* 4 *leds and the green cross represents the fixation point.*

5.2 Experimental setup

In this section, the different parameterizations chosen to conduct the experiments on the dataset introduced above will be briefly reviewed. First, concerning the preprocessing then for the differents steps composing the Riemannian classification pipeline. Finally, at the level of the performance measures used.

5.2.1 Preprocessing

There are countless ways to preprocess signals, but there is no universal procedure. The best way to proceed is on a case-by-case basis. Regarding our dataset, we will first extend our signals and then extract the trials, as suggested in [Kal+16].

Signals extension In this dataset, SSVEP signals are used. For these types of signals, the filtered signal covariance is advised [YBL17][CBA13]. The principle is to extend the original signal to include the filtered signal around each stimulation frequency:

Let $X \in \mathbb{R}^{C \times N}$ be the multi-channel signals, we define

$$X_{ext} = \begin{bmatrix} X_{freq_1} \\ \vdots \\ X_{freq_F} \end{bmatrix} \in \mathbb{R}^{FC \times N},$$

where C is the number of channels, N the number of samples and F the number of stimulation frequencies.

To obtain the filtered signals $(X_{freq_1}, \ldots, X_{freq_F})$, we use a 5th order bandpass Butterworth filter around the stimulation frequency.

Signals truncation In [Kal+16], they noticed that after cue onset $\tau_0 = 0$ the signal can still be synchronized with the previous trial frequencies. It is therefore crucial to wait after cue onset for the signals to be synchronized with the correct trial frequency.

For their signals, whose total duration of exposure to the stimulus is 5 seconds, they have demonstrated that taking into account only the last 3 seconds of the signal significantly increases, by 10%, the classification accuracy. We will do the same.

5.2.2 Classification pipeline

Following the Riemannian pipeline, several experiments will be conducted to compare the different classification methods and metrics.

Feature extraction For the Riemannian pipeline, the feature extraction step consists in estimating the EEG-signals covariance matrices.

For this purpose two estimators, introduced in Chapter 3, will be used: the Sample covariance matrix estimator (SCM) and the Shrunk covariance matrix estimator with the Ledoit and Wolf shrinkage.

Classification methods Regarding the selected classification methods, the two methods presented in Chapter 4 will be used, i.e. the Minimum Distance to the Mean (MDM) and the k Nearest Neighbors (k-NN).

Metrics In total three metrics, introduced in Chapter 2, are compared: the Euclidian metric and two Riemannian metrics, the Affine Invariant Riemannian metric (AIRM) and the Wassertein metric.

5.2.3 Performances measures

Based on the performances measures introduced in Chapter 4, we decided to keep the following for the three different levels of comparison.

Classification accuracy Concerning the performances measures we will use to compare our methods and metrics, we will measure the classification accuracy using the macro-F1 instead of the micro-F1 because it considers each class equally. We also need to choose which validation method we will use to estimate the accuracy measures. Since the dataset

is quite small, the leave-one-out approach can be performed in a reasonable amount of time.

Computation time The final goal of the experiment is to drive a wheelchair, for that reason, we decided that the most relevant time to take into account was the classification time of a new signal which will define the reaction time of the wheelchair and so the time the user needs to wait for his wheelchair to take the correct path. We do not take into account the time needed for training the classifier, e.g., computing the mean of each class for MDM, since the computation is done offline and is therefore not critical in the workflow.

Memory usage Memory usage is analyzed from a theoretical point of view, i.e., in terms of asymptotic complexity. We only consider the memory needed for storing the classifier once it has been trained. Factors that impact this memory complexity are the number of channels, the number of frequencies, the chosen rank, and the classification method.

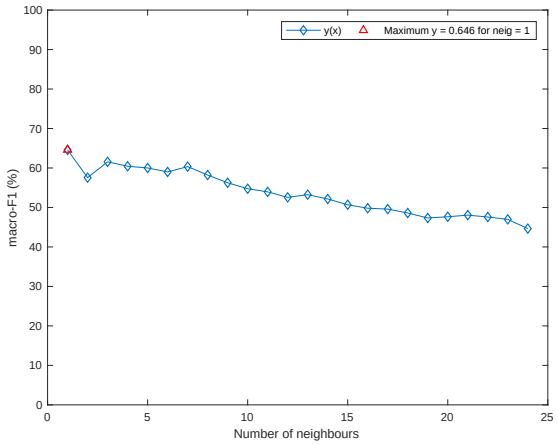
5.3 Hyperparameter optimization

The k-NN method, as can be seen from its name, has a hyperparameter k that we must set a priori. In order not to fix a value randomly, we can use the previously mentioned cross-validation to find the optimal value of this parameter according to the dataset, the method and the distance used.

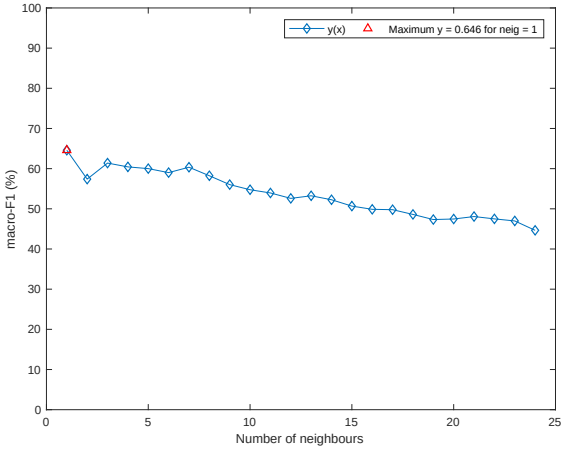
The principle is the following: we simply compute the classification macro-F1 score for each subject for each possible number of neighbors k (from 1 to 24) and then make an average of all subjects for each number of neighbors. Those results, obtained for each of the three metrics and for two covariance estimators (sample covariance and Ledoit-Wolf) are represented in figure Fig. 5.3.

Taking the maximum macro-F1, with the SCM estimator we can set respectively for the Euclidean, affine-invariant and full-rank Wasserstein metrics: $k = 1$, $k = 19$, $k = 1$. Similarly, with the Ledoit-Wolf estimator, we can set respectively for the Euclidean, affine-invariant and full-rank Wasserstein metrics: $k = 1$, $k = 15$, $k = 1$.

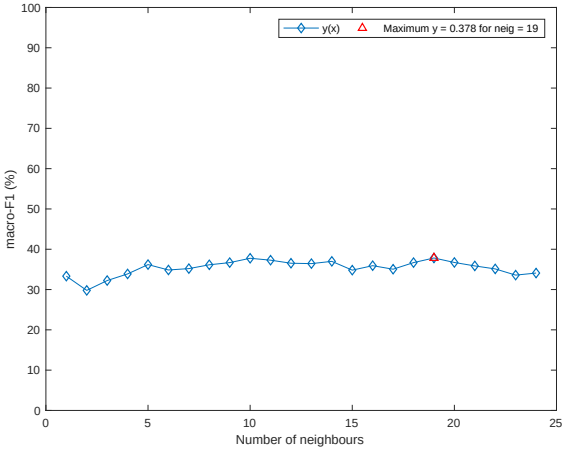
For both estimators, we can clearly see that the curves of the Euclidean and the Wasserstein metrics are strictly decreasing. There is no doubt that the best neighbor number is equal to 1. A possible interpretation of this result, at least for the Euclidean metric, is that the farther away a neighbor is, the more defective the distance measure is. For the AIRM metric, the trend of the curve is more difficult to detect. In any case, it seems to be relatively constant. This means that even if 19 and 15 are respectively the best number of neighbors for the AIRM metric with MDM and k-NN methods for this dataset, other values are also good candidates.



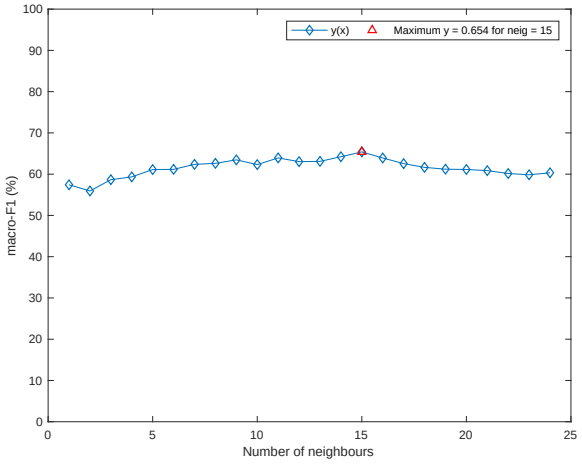
(a) Euclidean, SCM, $k = 1$



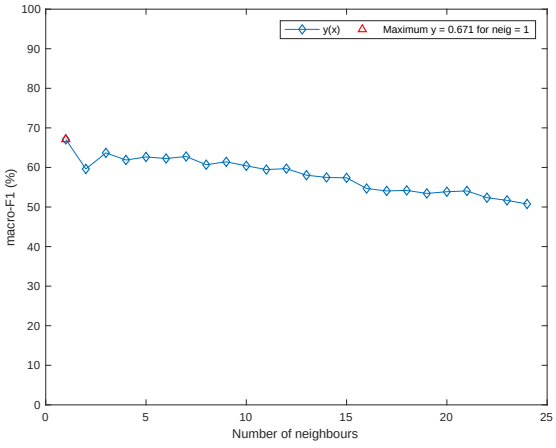
(b) Euclidean, LW, $k = 1$



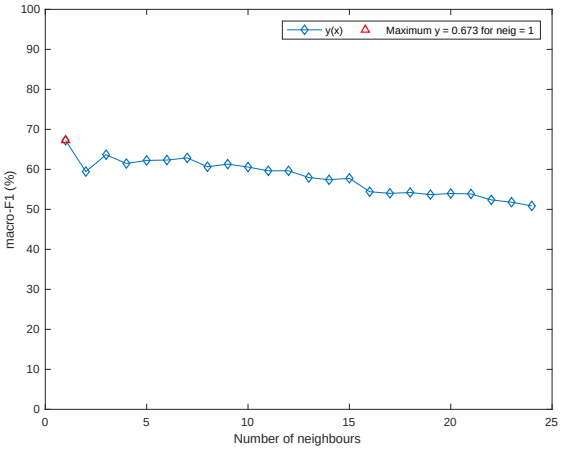
(c) AIRM, SCM, $k = 19$



(d) AIRM, LW, $k = 15$



(e) Wass, SCM, $k = 1$



(f) Wass, LW, $k = 1$

Figure 5.3: Evolution of the macro-F1 classification score for different numbers of neighbors k for the k-NN method. On the left, covariance was estimated using sample covariance (SCM), on the right with Ledoit-Wolf shrinkage (LW). Each row corresponds to a specific metric. Below each case we write the optimal choice of k.

5.4 Impact of covariance estimators

In Chapter 3 we introduced estimators for covariance matrices. In this section, we study how these estimators influence the classification performance of our two methods (MDM and k-NN) combined with our three metrics (Euclidean, AIRM and Wasserstein). Based on the previously introduced shrunk covariance:

$$\boldsymbol{\Sigma}_{\text{shrink}} = (1-\rho)\boldsymbol{\Sigma}_{scm} + \rho\nu\mathbf{I},$$

where ρ is a coefficient to be chosen and $\nu = \text{tr}(\boldsymbol{\Sigma}_{scm})/n$ the average eigenvalue of the sample covariance matrix.

We compare the three following estimators:

- The sample covariance estimator (SCM), with $\rho = 0$,
- The shrunk covariance estimator with a fixed shrinkage ρ,
- The shrunk covariance with the Ledoit-Wolf shrinkage estimator (LW) which computes for each signal the optimal shrinkage ρ_{LW} to use.

5.4.1 Classification accuracy

For each of the three covariance estimators, we want to analyze how they influence the F1-macro score with each of the two classification methods and each of the three metrics. To do that, we compute for each subject the F1-macro score for several different shrinkage values. Then we compute the mean over the subjects.

Minimum Distance to the Mean For the MDM method, we obtained the graph represented in figure 5.4. The performance achieved using the SCM estimator corresponds to the leftmost points on the plot. For each metric, we used different fixed shrinkage values to compute the shrunk covariances. We can observe that for the Euclidean and the Wasserstein metrics, the shrinkage does not influence the macro-F1 score except for large shrinkage values (≥ 0.5). Though, their classification's accuracy are very different: using the Wasserstein metric allow us to reach a macro-F1 score 10% higher than the one achieved with the Euclidean metric. However, with the AIRM metric, the shrinkage has a positive influence on the performance: maximal performance is achieved around $\rho = 0.01$ and beats the other two metrics.

The dashed lines correspond to the macro-F1 score reached when using the Ledoit-Wolf shrinkage estimator, which is considered as optimal. For the Euclidean metric and the Wasserstein metric, those values are the same as the ones obtained with the two previous estimators, confirming that the classification accuracy with those metrics is not greatly influenced by the shrinkage. Concerning the AIRM metric, the macro-F1 score obtained with the Ledoit-Wolf shrinkage estimator is slightly higher than the maximum reached by fixing the shrinkage value which also confirm that the Ledoit-Wolf method to compute the shrinkage is optimal.

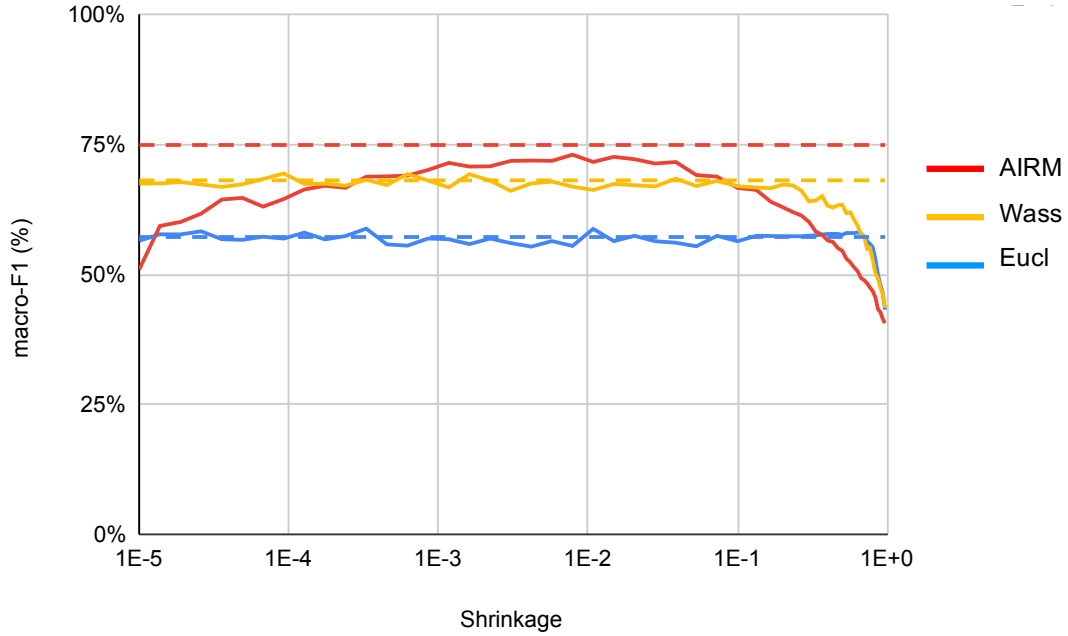


Figure 5.4: Macro-F1 evolution for different values of shrinkage with the MDM classification method. Dotted lines represent the optimal macro-F1 scores when the shrinkages are chosen by the Ledoit-Wolf estimator for each subject.

The results shown figure 5.4 raise the following question: is the Ledoit-Wolf shrinkage calculated for each covariance matrix optimal? In other words, by multiplying this shrinkage by a multiple κ, could we obtain a better classification performance? To analyze this, we reconduct the same experiment by setting $\rho = \rho_{LW} * \kappa$ and look at the influence of κ on the classification performance. The results are represented figure 5.5. As before, we observe that the Euclidean and Wasserstein metric are practically independent of the shrinkage parameter. On the other hand, the affine-invariant metric is optimal for $0.5 \leq \kappa \leq 2$, and in this range it is practically equivalent to $\kappa = 1$. We can safely conclude from this experiment that the Ledoit-Wolf shrinkage is indeed the best possible parameter for improving the classification performance.

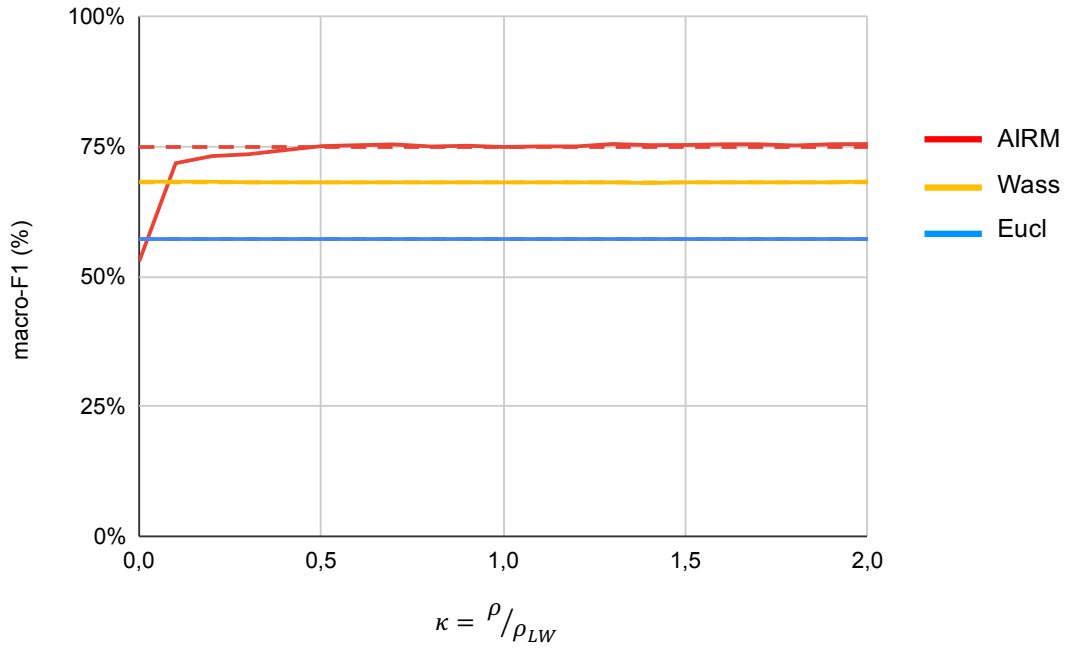


Figure 5.5: Macro-F1 evolution for different values of κ with the MDM classification method. Dotted lines represent the optimal macro-F1 scores when the shrinkages are chosen by the Ledoit-Wolf estimator for each subject.

k-Nearest Neighbors For the k-NN method, we obtained the graph represented figure 5.6. This graph was generated in the same way as the one for the MDM but has a very different look. First, it is important to notice that none of the metrics reach the macro-F1 score of 75% unlike in MDM. Furthermore, here even if they do not depend on the chosen shrinkage as observed in MDM, the Euclidean and the Wasserstein metrics have practically the same classification accuracy. Concerning the AIRM metric, the previous observations are also correct for the k-NN method: the classification accuracy is greatly influenced by the shrinkage's choice.

When using the Ledoit-Wolf shrinkage estimator the findings are similar to those obtained previously. However, the only difference is that the AIRM does not outperform the two other metrics anymore. They all reach the same classification accuracy.

In terms of classification accuracy, we can draw the following conclusions:

- For the AIRM metric, the use of a shrinkage is crucial; it allows the metric to outperform the others when choosing the right shrinkage with the MDM classification method.
- For the two other metrics, the Euclidean and the Wasserstein metrics, the use of a shrinkage does not impact the classification performances at all.
- Using the SCM estimator, the best classification accuracy is reached with the Wasserstein metric and the MDM method.

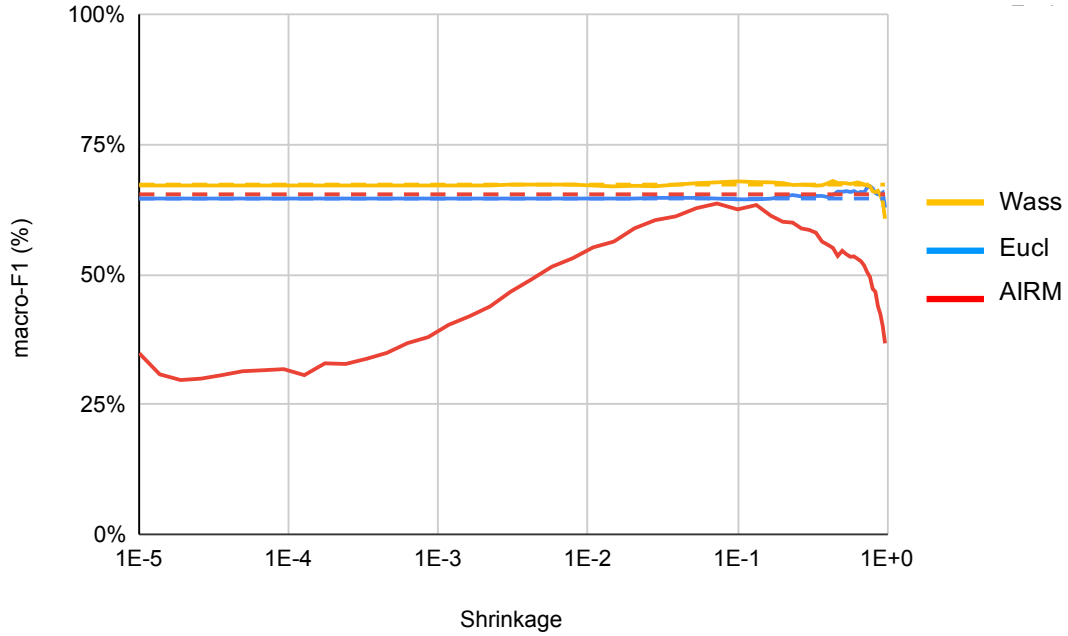


Figure 5.6: Macro-F1 evolution for different values of shrinkage with the k-NN classification method. Dotted lines represent the optimal macro-F1 scores when the shrinkages are chosen by the Ledoit-Wolf estimator for each subject.

- Using the Ledoit-Wolf estimator that chooses the right shrinkage value for each covariance matrix with the AIRM metric and the MDM method, we reach the best classification accuracy. Ledoit-Wolf is the optimal covariance estimator for the AIRM metric.
- The k-NN method does not bring any improvement compared to the MDM method.

5.4.2 Computation time

In order to have an idea of the computation time needed to process a new signal, we timed the two main parts of the classification of a new signal: the estimation of its covariance matrices and the classification of these. To do this, we measure the average times that these two calculations take and we obtained the graph on Figure 5.7.

Regarding the covariance estimators, we can see that the calculation times are quite similar between the two estimators but the best is the SCM estimator since it does not require to compute an optimal shrinkage coefficient.

Regarding the method and the metric used, we can first notice that the times obtained with the MDM classification method are significantly smaller than the ones obtained with the k-NN classification method. This observation makes sense regarding the respective methods' way of working: k-NN requires to compute the distance to all matrices in the dataset, whereas MDM only computes the distance to the center matrices (one per class).

We also observe that between them, the three metrics have very different computation times. The Euclidean one is by far the quickest, followed by the AIRM metric and then

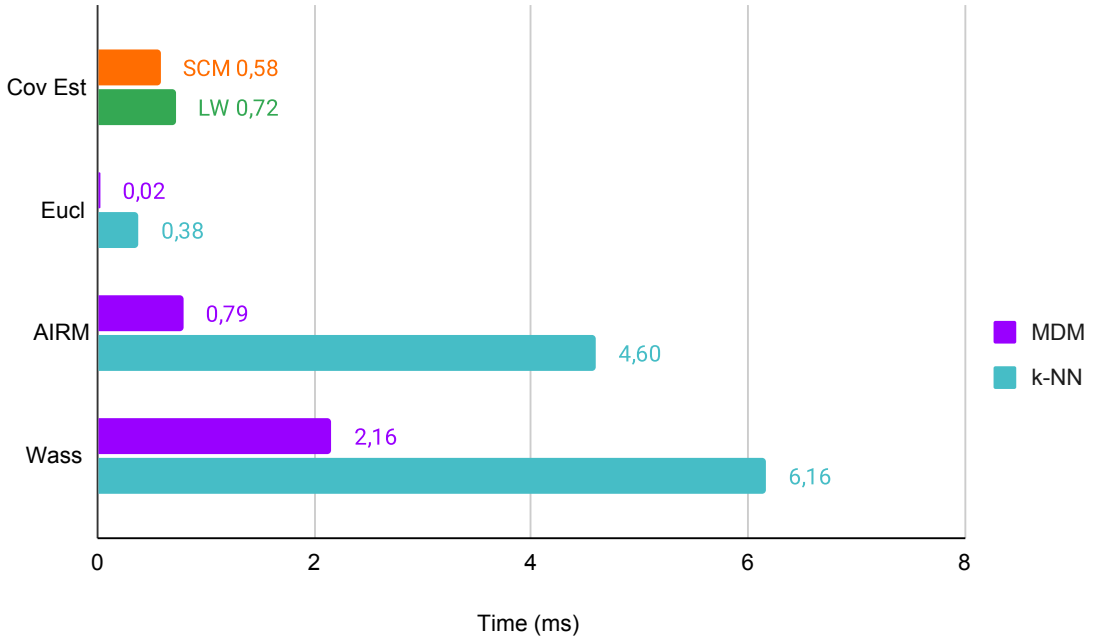


Figure 5.7: Computation times for one sample classification with several possible covariance estimators (Cov Est), upper part, and classification methods and metrics, lower part.

the Wasserstein metric. However, it is important to note that the computation times obtained are very reasonable, of the order of a few milliseconds.

Taking into account both parts of the graph, we can see that the computation time needed for the computation of the covariance matrices with the help of an estimator is small but not negligible in the classification process.

In terms of computation time, the best covariance estimator is the SCM estimator and the best method-metric combination the Euclidean metric with the MDM classifier.

5.4.3 Memory usage

We are interested in the memory space needed to store the classifier. Let us define F, the number of frequencies, C, the number of channels and N, the number of time samples per trials. The extended signals have FC channels, hence the covariance matrices have dimensions $FC \times FC$.

In the general case where we consider full rank matrices:

- The MDM classification method needs to store the mean matrices i.e. $F+1$ matrices, one per frequency and the resting class, of size $FC \times FC$. The memory complexity is thus $\mathcal{O}(F^3C^2)$.
- The k-NN classification method needs to store all the matrices i.e. N matrices, one per sample/trial, of size $FC \times FC$. The complexity is thus $\mathcal{O}(NF^2C^2)$.

In terms of memory usage, since $N \gg F$, the MDM method is better because it uses less memory. Taking into account the use of covariance matrix' estimators, the conclusion

remains the same. Indeed, what we are interested in here as memory usage is the long-term storage whereas the computation of these covariance matrices using estimators will solicit another memory, the random access memory, which we do not consider here.

5.5 Impact of low-rank metrics

In this section we investigate the performance of classifiers using the low-rank Wasserstein metric. First, the spectrum of the matrices of the dataset is analyzed to justify the low-rank approximation. Then we examine the impact of the chosen fixed rank R on classification accuracy, computation time and memory usage.

5.5.1 Low-rank nature of the matrices

To analyze the spectrum of our covariance matrices, we compute the eigenvalues of each covariance matrix for a subject picked randomly, here the first subject. The eigenvalues are sorted in descending order. We represent the distribution of each eigenvalue with a box plot.

We obtained the graph on Figure 5.8. We can directly notice that the matrix energy is mainly concentrated in the first eigenvalues. It is reasonable to assume that the remaining eigenvalues represent only noise. It is therefore probably a good idea to ignore them during classification.

Given these observations, it makes sense to approximate these covariance matrices by low-rank covariance matrices. We must therefore use methods that can correctly handle this low-rank nature. For this purpose, we can re-use the two classification methods previously used, MDM and k-NN, combined with the Wasserstein metric which will allow us to treat low-rank matrices.

5.5.2 Classification accuracy

To analyze how the covariance matrices rank influences the classification accuracy, we compute for each subject the macro-F1 score obtained using the low-rank approximation covariance matrices with the SCM estimator. Then we compute the mean over the subjects. The obtained graphs are represented on Figures 5.9 and 5.10, respectively with the MDM and the k-NN classification methods.

Minimum Distance to the Mean For the MDM method, we can notice a increasing curve that stabilizes approximately around rank 8. The maximal value for the macro-F1 score is obtained at the 13rd rank. However, the curve is almost constant after rank 8 which means that from this rank, we will not gain in precision by increasing the rank and it is thus not necessary to consider the full rank matrix. This makes sense with Figure 5.8 previously showed. We can see that from the 8th eigenvalue, the remaining eigenvalues are negligible.

k-Nearest Neighbors For the k-NN method, the observation is almost the same. Except that the curve seems to be less increasing and stabilizes even faster than the one

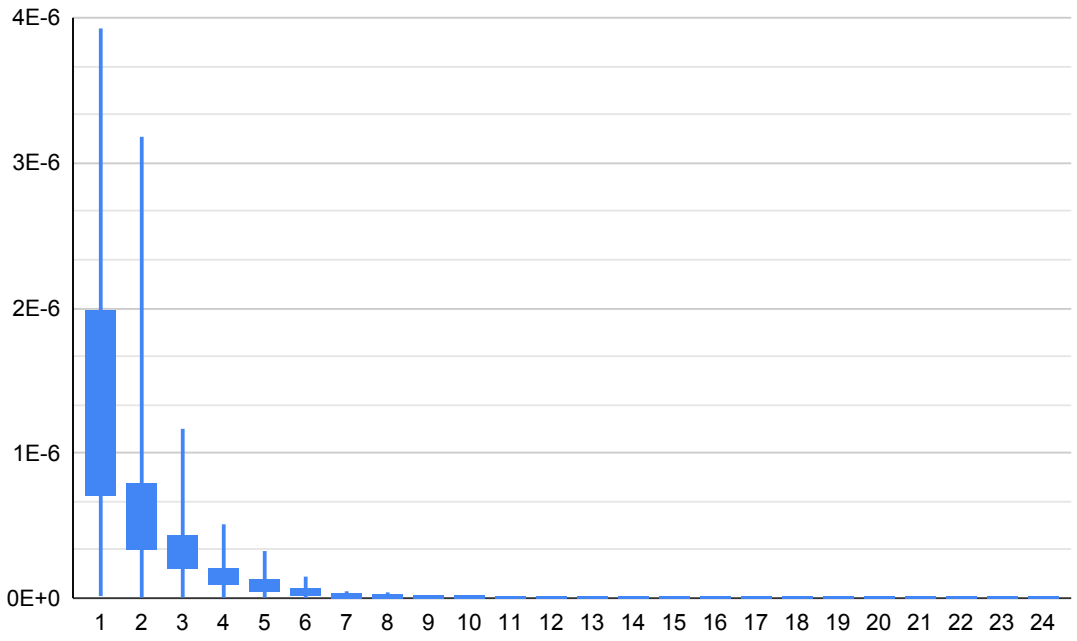


Figure 5.8: Eigenvalues' spectrum distribution for the 24 eigenvalues of the first subject.

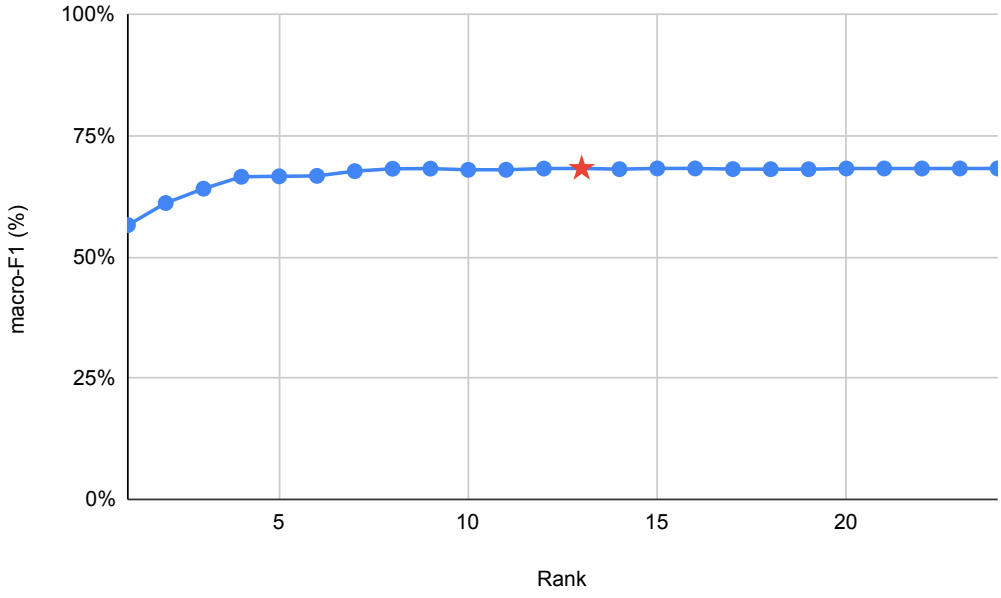


Figure 5.9: Macro-F1 evolution for different ranks with the MDM classification method. The red star represents the optimal macro-F1 scores.

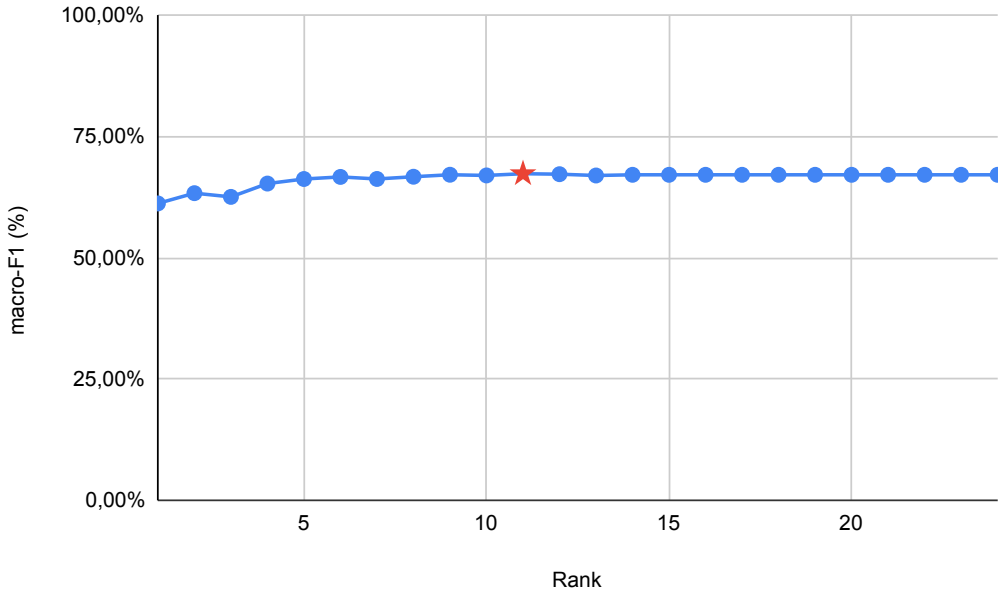


Figure 5.10: Macro-F1 evolution for different ranks with the k-NN classification method. The red star represents the optimal macro-F1 scores.

obtained for the MDM, already from rank 5 and that the macro-F1 score maximal value is reached with rank 11.

In terms of classification accuracy, the conclusion is the same for both methods: the performance achieved by the full rank covariance matrices can already be achieved with the low-rank approximations of these covariance matrices. Therefore, using low-rank approximations does not lead to a loss of classification accuracy if the rank is correctly chosen.

5.5.3 Computation time

What interests us here is to see how the computation time evolves as a function of the rank of the covariance matrix approximation. To do this, we compute the mean time needed to classify a new sample.

Minimum Distance to the Mean For the MDM method, we obtained the graph represented in figure 5.11. We can observe that the rank greatly influences the computation time. This shows that one can reduce the computation time by a factor 3 or 4 while preserving the classification accuracy.

***k*-Nearest Neighbors** For the k-NN method, we obtained the graph represented in figure 5.12. The observations are similar to those observed for the MDM except that the computation time is much higher in this case.

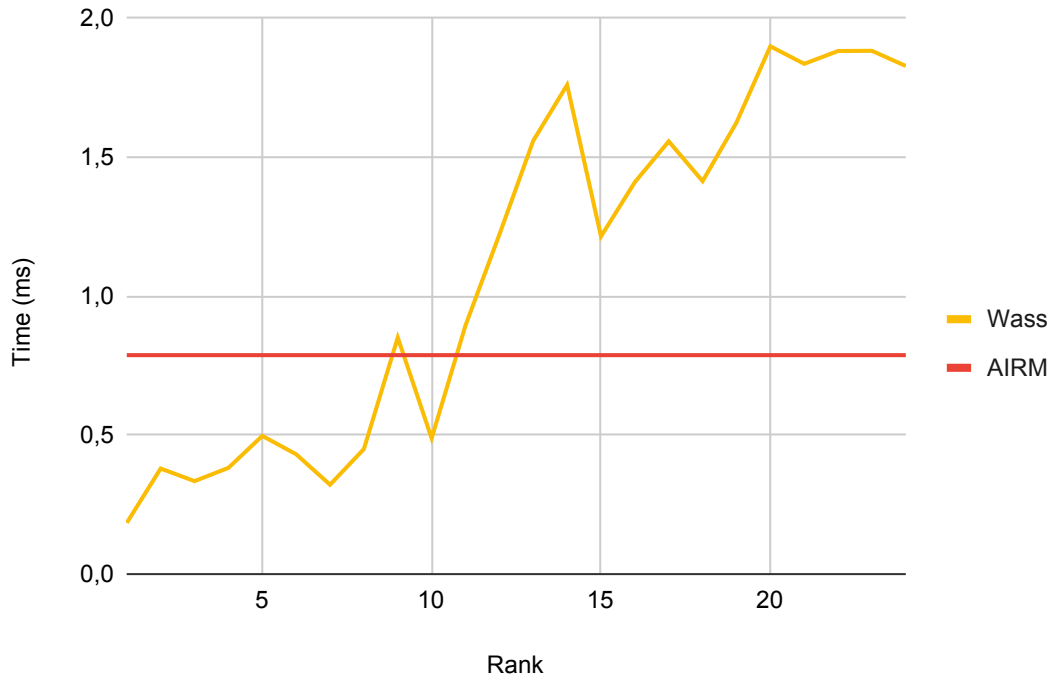


Figure 5.11: Computation time evolution for each rank with the MDM classification method and the Wasserstein metric. The red line represents the computation time for one sample classification with the AIRM metric.

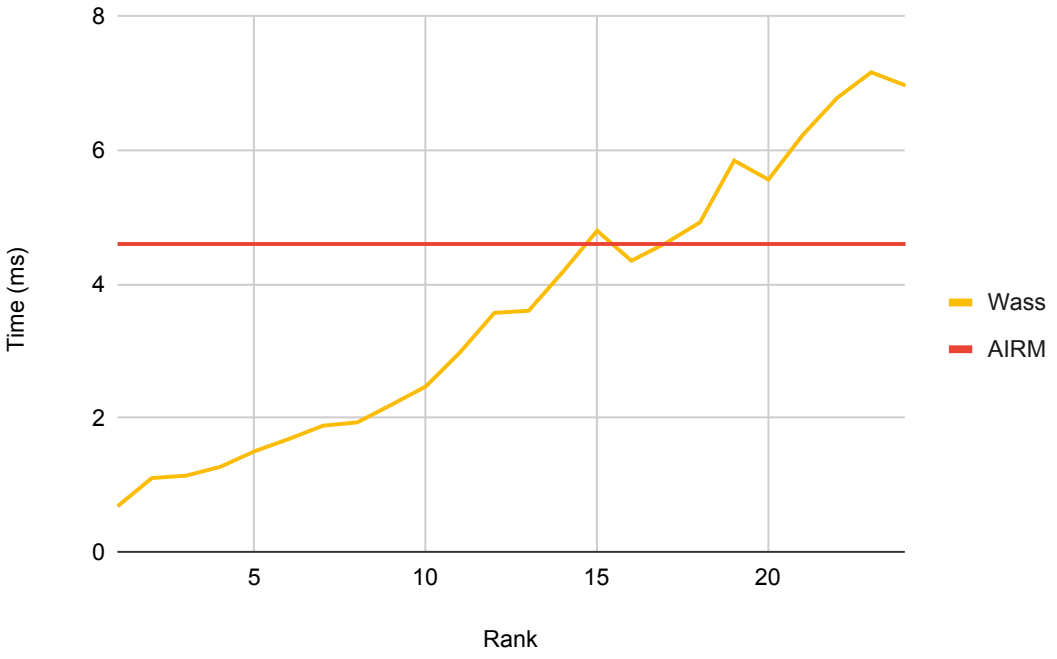


Figure 5.12: Computation time evolution for each rank with the k-NN classification method and the Wasserstein metric. The red line represents the computation time for one sample classification with the AIRM metric.

5.5.4 Memory usage

The fact of reducing the rank of the matrices will have an impact on the memory space needed to store the classifier. In fact, the new low-rank covariances matrices have now the following dimensions $FC \times R$ where R is the chosen rank.

If we consider those low-rank matrices:

- The MDM classification method needs to store the mean matrices i.e. $F+1$ matrices, one per frequency and the resting class, of size $FC \times R$. The memory complexity is thus $\mathcal{O}(F^2CR)$.
- The k-NN classification method needs to store all the matrices i.e. N matrices, one per sample/trial, of size $FC \times R$. The complexity is thus $\mathcal{O}(NFCR)$.

Since $R \leq FC$, both classification methods will use less memory space than the full-rank. Moreover, the lower the rank, the higher the memory gain. For much larger datasets, i.e. with many channels, this can play a decisive role.

5.6 Conclusions

From the many results obtained, we can draw the following main conclusions:

- The Minimum Distance to the Mean classifier offers better performances than k-Nearest Neighbors, in terms of accuracy, time and memory. k-NN should therefore never be chosen in a practical context.
- Using the Ledoit-Wolf shrunk covariance instead of the standard sample covariance, can significantly improve the classification accuracy (at worst, it offers similar performance), with very little overhead in terms of computation time.
- While the AIRM metric slightly outperforms the Wasserstein metric with the right covariance estimator, the Wasserstein metric is practically independent of the choice of the covariance estimator, and is therefore more robust.
- When using the low-rank Wasserstein metric with a properly chosen rank, one can maintain good accuracy while significantly decreasing the time and memory costs of the classifier.

Chapter 6

Conclusion and Future Work

In this work, we have analyzed and compared different Riemannian metrics for classifying EEG signals. Our contribution is the use of the recent Wasserstein metric on the manifold of fixed-rank positive semidefinite matrices, introduced by [MA20]. Instead of computing distances from the full covariance matrices, we leverage the low-rank nature of EEG-based covariance matrices: we keep only a low-rank approximation, and measure distances on the low-rank manifold. This work has shown two main results:

- For comparable experiments, i.e., experiments using the same classification method, distance function, validation method and covariance estimator, the classification performance achieved by the Wasserstein metric is comparable to the one obtained by state-of-the-art, full-rank Riemannian metrics. Moreover, if the rank is wisely chosen, the Wasserstein metric is faster and less expensive in terms of memory storage than the state-of-the-art.
- The Wasserstein metric is practically insensitive to the value of the shrinkage for a wide range of values and in particular for ill-conditioned matrices, i.e. when the shrinkage is close to zero. Contrary to the standard affine-invariant metric which does not correctly handle ill-conditioned matrices.

Based on these observations, one can expect that for large matrices, which naturally become ill-conditioned, the Wasserstein metric outperforms the affine-invariant metric in terms of classification performance. Indeed, we expect the performance of the affine-invariant metric to degrade with the size of the matrices. In addition to that, the Wasserstein metric guarantees significant time and memory savings thanks to its low-rank approximations.

These expectations deserve to be studied in more detail and open new perspectives:

- Testing the Wasserstein metric on high-dimensional datasets, such as [Nik21], would allow us to confirm our assumptions. This track is therefore a high priority.
- The algorithm for computing the means with the Wasserstein metric is not optimized. In order to save computation time, it would be interesting to optimize it and to develop faster and more stable methods such as the Karcher average for the affine-invariant distance.

- The classification methods used in this work, i.e., Minimum Distance to the Mean and k Nearest Neighbors, remain basic. It would therefore be interesting to use more sophisticated classification methods, such as the kernel-based Support Vector Machine, with the Wasserstein metric in order to achieve better classification performance and compare the obtained results with the state-of-the-art.

Of course, this list of research directions is by no means exhaustive and there are still many challenges that will have to be overcome to allow BCIs to be used outside laboratory conditions. However, there is no doubt that BCIs and all the disciplines on which they rely on, thanks to their fast-moving but also human-oriented nature, continue to attract and fascinate many researchers eager to put their knowledge at the service of such a promising field.

Bibliography

[AMS09] P.-A. Absil, R. Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Apr. 2009. DOI: 10.1515/9781400830244.

[Bar+12] A. Barachant et al. "Multiclass Brain–Computer Interface Classification by Riemannian Geometry". In: *IEEE Transactions on Biomedical Engineering* 59.4 (Apr. 2012), pp. 920–928. DOI: 10.1109/TBME.2011.2172210.

[Bar+13] Alexandre Barachant et al. "Classification of covariance matrices using a Riemannian-based kernel for BCI applications". en. In: *Neurocomputing* 112 (July 2013), pp. 172–178. DOI: 10.1016/j.neucom.2012.12.039.

[BI13] Dario A. Bini and Bruno Iannazzo. "Computing the Karcher mean of symmetric positive definite matrices". en. In: *Linear Algebra and its Applications* 438.4 (Feb. 2013), pp. 1700–1710. DOI: 10.1016/j.laa.2011.08.052.

[BJL19] Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. "On the Bures–Wasserstein distance between positive definite matrices". en. In: *Expositiones Mathematicae* 37.2 (June 2019), pp. 165–191. DOI: 10.1016/j.exmath.2018.01.002.

[Bla+11] Benjamin Blankertz et al. "Single-Trial Analysis and Classification of ERP Components — A Tutorial". In: *NeuroImage* 56.2 (May 2011), pp. 814–825. DOI: 10.1016/j.neuroimage.2010.06.048.

[Bou+14] N. Boumal et al. "Manopt, a Matlab toolbox for optimization on manifolds". In: *Journal of Machine Learning Research* 15.42 (2014), pp. 1455–1459.

[BS20] Sándor Beniczky and Donald L. Schomer. "Electroencephalography: Basic Biophysical and Technological Aspects Important for Clinical Applications". In: *Epileptic Disorders: International Epilepsy Journal with Videotape* 22.6 (Dec. 2020), pp. 697–715. DOI: 10.1684/epd.2020.1217.

[CBA13] Marco Congedo, Alexandre Barachant, and Anton Andreev. "A New Generation of Brain-Computer Interface Based on Riemannian Geometry". In: *arXiv:1310.8115 [cs, math]* (Oct. 2013).

[Che21] Sylvain Chevallier. *SSVEP Exoskeleton Dataset*. https://github.com/sylvchev/dataset-ssvep-exoskeleton. Sept. 2021.

[Dav16] Robert Davie. *Manifolds*. https://youtu.be/w1FaAu0SL-w. 2016.

[JB18] Vinay Jayaram and Alexandre Barachant. "MOABB: Trustworthy Algorithm Benchmarking for BCIs". In: *Journal of Neural Engineering* 15.6 (Dec. 2018), p. 066011. DOI: 10.1088/1741-2552/aadea0.

[Kal+16] Emmanuel K. Kalunga et al. "Online SSVEP-based BCI using Riemannian geometry". en. In: *Neurocomputing* 191 (May 2016), pp. 55–68. DOI: 10.1016/j.neucom.2016.01.007.

[LBC15] Fabien Lotte, Laurent Bougrain, and Maureen Clerc. "Electroencephalography (EEG)-Based Brain-Computer Interfaces". en. In: *Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons, Inc., Sept. 2015, pp. 1–20. DOI: 10.1002/047134608X.W8278.

[Lee18] John M. Lee. *Introduction to Riemannian Manifolds*. Vol. 176. Graduate Texts in Mathematics. Cham: Springer International Publishing, 2018. DOI: 10.1007/978-3-319-91755-9.

[Lot+07] Fabien Lotte et al. "A Review of Classification Algorithms for EEG-based Brain–Computer Interfaces". In: *Journal of Neural Engineering* 4 (2007), p. 24. DOI: 10.1088/1741-2560/4/2/R01.

[Lot+18] F Lotte et al. "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update". In: *Journal of Neural Engineering* 15.3 (June 2018), p. 031005. DOI: 10.1088/1741-2552/aab2f2.

[LW04] Olivier Ledoit and Michael Wolf. "A well-conditioned estimator for large-dimensional covariance matrices". en. In: *Journal of Multivariate Analysis* 88.2 (Feb. 2004), pp. 365–411. DOI: 10.1016/S0047-259X(03)00096-4.

[MA20] Estelle Massart and P.-A. Absil. "Quotient Geometry with Simple Geodesics for the Manifold of Fixed-Rank Positive-Semidefinite Matrices". en. In: *SIAM Journal on Matrix Analysis and Applications* 41.1 (Jan. 2020), pp. 171–198. DOI: 10.1137/18M1231389.

[Moa05] Maher Moakher. "A Differential Geometric Approach to the Geometric Mean of Symmetric Positive-Definite Matrices". en. In: *SIAM Journal on Matrix Analysis and Applications* 26.3 (Jan. 2005), pp. 735–747. DOI: 10.1137/S0895479803436937.

[Nag19] Sebastian Nagel. "Towards a Home-Use BCI: Fast Asynchronous Control and Robust Non-Control State Detection". Doctoral dissertation. Dec. 2019. DOI: 10.15496/publikation-37739.

[Nik21] Spiros Nikolopoulos. *MAMEM EEG SSVEP Dataset I (256 Channels, 11 Subjects, 5 Frequencies Presented in Isolation)*. 2021. DOI: 10.6084/M9.FIGSHARE.2068677.V6.

[NNL18] Chang S. Nam, Anton Nijholt, and Fabien Lotte, eds. *Brain-computer interfaces handbook: technological and theoretical advances*. Taylor & Francis, CRC Press, 2018. DOI: 10.1201/9781351231954.

[Pis+14] I. Pisarenco et al. "High-density electroencephalography as an innovative tool to explore sleep physiology and sleep related disorders". en. In: *International Journal of Psychophysiology* 92.1 (Apr. 2014), pp. 8–15. DOI: 10.1016/j.ijpsycho.2014.01.002.

[Ras+20] Mamunur Rashid et al. "Current Status, Challenges, and Possible Solutions of EEG-Based Brain-Computer Interface: A Comprehensive Review". In: *Frontiers in Neurorobotics* 14 (2020), p. 25. DOI: 10.3389/fnbot.2020.00025.

[Sab+19] David Sabbagh et al. "Manifold-regression to predict from MEG/EEG brain signals without source modeling". In: *arXiv:1906.02687 [cs, eess, stat]* (Nov. 2019).

[Shm20a] Boaz Shmueli. *Multi-Class Metrics Made Simple, Part I: Precision and Recall*. en. https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2. June 2020.

[Shm20b] Boaz Shmueli. *Multi-Class Metrics Made Simple, Part II: the F1-score*. en. https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1. July 2020.

[SS05] Juliane Schäfer and Korbinian Strimmer. "A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics". In: *Statistical Applications in Genetics and Molecular Biology* 4.1 (Jan. 2005). DOI: 10.2202/1544-6115.1175.

[Ste56] Charles Stein. "Inadmissibility of the usual estimator for the mean of a multivariate normal distribution". In: *Contribution to the Theory of Statistics*. University of California Press, Dec. 1956, pp. 197–206. DOI: 10.1525/9780520313880-018.

[Szy11] Brett Szymik. *What's Your Brain Doing?* https://askabiologist.asu.edu/brain-regions. May 2011.

[TPM08] O. Tuzel, F. Porikli, and P. Meer. “Pedestrian Detection via Classification on Riemannian Manifolds”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10 (Oct. 2008), pp. 1713–1727. DOI: 10.1109/TPAMI.2008.75.

[Vid+09] Carmen Vidaurre et al. “Time Domain Parameters as a Feature for EEG-based Brain–Computer Interfaces”. In: *Neural Networks* 22.9 (Nov. 2009), pp. 1313–1319. DOI: 10.1016/j.neunet.2009.07.020.

[Wik21] Wikipedia contributors. *Tangent space — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Tangent_space&oldid=1061243268. 2021.

[WW12] Jonathan R. Wolpaw and Elizabeth Winter Wolpaw, eds. *Brain-computer interfaces: principles and practice*. Oxford ; New York: Oxford University Press, 2012. DOI: 10.1093/acprof:oso/9780195388855.001.0001.

[YBL17] Florian Yger, Maxime Berar, and Fabien Lotte. “Riemannian Approaches in Brain-Computer Interfaces: A Review”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.10 (Oct. 2017), pp. 1753–1762. DOI: 10.1109/TNSRE.2016.2627016.

[YLS15] Florian Yger, Fabien Lotte, and Masashi Sugiyama. “Averaging covariance matrices for EEG signal classification based on the CSP: An empirical study”. In: *2015 23rd European Signal Processing Conference (EUSIPCO)*. Nice: IEEE, Aug. 2015, pp. 2721–2725. DOI: 10.1109/EUSIPCO.2015.7362879.

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