UNIVERSITÉ PARIS SCIENCES & LETTRES

Thèse de doctorat
de l'Université PSL

Préparée à l'Université Paris Dauphine - PSL

Wrapping Around the Manifold: Geometric Statistics for Brain-Computer Interfaces

Soutenue par

Thibault de SURREL de SAINT JULIEN

Le 6 Novembre 2026

École doctorale n°543

SDOSE

Spécialité

Informatique

Composition du jury :

Gabriel PEYRÉ Directeur de recherche, CNRS, École Normale Supérieure	*Président du jury*
Stéphane BONNET Ingénieur de Recherche, CEA Centre de Grenoble	*Rapporteur*
Xavier PENNEC Directeur de Recherche, Inria de Sophia-Antipolis	*Rapporteur*
Estelle MASSART Assistant professor, UCLouvain	*Examinatrice*
Michael TANGERMANN Associate professor, Radboud University	*Examinateur*
Sylvain CHEVALLIER Professeur, LISN - Université Paris-Saclay	*Directeur de thèse*
Fabien LOTTE Directeur de Recherche Inria, Centre Inria de l'Université de Bordeaux / LaBRI	*Encadrant de thèse*
Florian YGER Maître de conférences, INSA Rouen-Normandie	*Encadrant de thèse*

Dauphine UNIVERSITÉ PARIS SCIENCES & LETTRES

To do things right, first you need love, then technique.

—Antoni Gaudí

The universe which we now enter is one of dimension at least 6 and this means that we will have trouble drawing meaningful pictures, not to mention keeping our calculations on small pieces of paper. The reader is advised to get some big sheets of paper to do some of the exercises (or maybe an account on a supercomputer).

—Audrey Terras (1988), Harmonic Analysis on Symmetric Spaces and Applications II

Abstract

Modern machine learning tools often rely on quantifying similarities between data points. When the data of interest are highly structured, using classical Euclidean geometry can lead to infeasible or suboptimal results. Thus, taking this structure into account is essential for building reliable and powerful tools.

A particular area of interest in this thesis is *Brain–Computer Interfaces* (BCIs), whose goal is to translate electrical brain signals into concrete commands. In this work, the features used are *covariance matrices* estimated from recorded electroencephalograms. These covariance matrices have a strong mathematical structure, as they are *Symmetric Positive Definite* (SPD) matrices. Over the past 15 years, algorithms based on the Riemannian geometry of the set $\mathcal{P}_d$ of SPD matrices have achieved state-of-the-art results in BCI classification tasks.

In this work, we focus on building a geometry-aware statistical model by mapping the Euclidean multivariate Gaussian distribution from the tangent space of the manifold $\mathcal{P}_d$ onto the manifold. This yields a *wrapped Gaussian* distribution. This probability distribution is studied theoretically, and the estimation of its parameters is discussed. More importantly, this distribution is used to build a probabilistic framework for machine learning on the manifold $\mathcal{P}_d$, encompassing existing classifiers, outlier detection methods, kernel methods, and even Riemannian flow matching methods. We also propose several extensions of these algorithms, taking advantage of the versatility and flexibility of the wrapped Gaussian distribution.

In the second part, we study the application of this statistical model to BCI by focusing on a specific problem. We investigate the detection of wakefulness during general anesthesia. To this end, we propose a new algorithm based on the wrapped Gaussian distribution, evaluate it on a dataset recorded during real surgeries, and show that this new anisotropic model is able to learn a more informative representation of the data than state-of-the-art methods.

Finally, we address the interpretability of Riemannian tools used in BCI, first by building a visualization method that adapts the t-SNE algorithm to the manifold $\mathcal{P}_d$, and then by assessing the contribution of each electrode to the classification results. This helps the different stakeholders of a BCI system, from BCI researchers to end users, better understand both the Riemannian geometry underlying these methods and the decisions made by Riemannian algorithms.

Résumé

Les outils modernes d'apprentissage automatique cherchent souvent à quantifier les similarités entre les points d'un jeu de données. Lorsque les données d'intérêt sont très structurées, utiliser la géométrie euclidienne classique peut produire des résultats incohérents ou sous-optimaux. De plus, prendre en compte cette structure est essentiel pour construire des outils puissants et fiables.

Un cas d'application particulier étudié dans cette thèse est celui des *Interfaces Cerveau-Ordinateur* (ICO), dont l'objectif est de traduire les signaux électriques du cerveau en commandes concrètes. Dans cette thèse, nous estimons des matrices de covariance à partir d'électroencéphalogrammes, et ce sont ces matrices de covariance qui servent de données d'entrée à nos algorithmes. Ces matrices de covariance sont mathématiquement très structurées, puisque ce sont des matrices *Symétriques Définies Positives* (SDP). Depuis une quinzaine d'années, les algorithmes exploitant la géométrie riemannienne de $\mathcal{P}_d$, l'espace des matrices SDP, figurent parmi les méthodes de l'état de l'art pour la classification en ICO.

Dans mes travaux, nous nous focalisons sur la construction d'un modèle statistique qui prend en compte la géométrie, en envoyant une gaussienne multivariée euclidienne d'un espace tangent de la variété $\mathcal{P}_d$ sur cette variété. Cela crée une *gaussienne enveloppée*. Cette distribution de probabilité est étudiée d'un point de vue théorique et l'estimation de ses paramètres est discutée. Mais surtout, cette distribution est utilisée pour créer un cadre probabiliste pour des outils d'apprentissage automatique sur la variété $\mathcal{P}_d$, englobant des outils connus comme des classifieurs, des outils de détection d'observations aberrantes, des méthodes à noyaux, mais aussi des méthodes de flots génératifs riemanniens. Nous proposons différentes extensions de ces algorithmes, en nous basant sur la polyvalence ainsi que la flexibilité de la gaussienne enveloppée.

Dans une deuxième partie, nous étudions une application de ce modèle statistique aux ICO en nous concentrant sur un problème en particulier. Nous nous attaquons à la détection de l'état d'endormissement lors d'une anesthésie générale. Pour cela, nous proposons un nouvel algorithme se basant sur les gaussiennes enveloppées, nous l'étudions sur un jeu de données spécifique et nous montrons que ce nouveau modèle anisotropique est capable d'apprendre une représentation plus informative des données que les algorithmes de l'état de l'art.

Enfin, nous abordons l'interprétabilité d'outils riemanniens utilisés en ICO, premièrement en construisant une méthode de visualisation qui adapte l'algorithme t-SNE à la variété $\mathcal{P}_d$, puis en évaluant les contributions de chaque électrode au résultat de classification. Cela aide les différentes parties prenantes des ICO, du chercheur en ICO à l'utilisateur final, à mieux appréhender la géométrie riemannienne ainsi que les décisions prises par les algorithmes riemanniens.

Acknowledgements

En premier lieu, j'aimerais remercier mes encadrants de thèse, Sylvain Chevallier, Fabien Lotte et Florian Yger. Merci de m'avoir fait confiance dès le stage de M2 et d'avoir accepté de continuer l'aventure en thèse avec moi. Merci pour votre écoute ainsi pour le temps que vous m'avez consacré, malgré vos emplois du temps très chargés. Je suis conscient de la chance que j'ai eue de vous avoir comme encadrants de thèse. Vous m'avez laissé la liberté d'explorer ce qui me passionnait vraiment, transformant un sujet initialement plus appliqué en un travail de recherche plus théorique. Je me suis très rapidement senti en confiance, libre d'avancer et de proposer mes propres idées, tout en sachant que vous me suivriez.

Sylvain, merci pour ton aide dans la rédaction des introductions pour les grandes conférences de ML, pour avoir pris le temps de discuter à de nombreuses reprises de la suite de ma carrière, ainsi que pour tes nombreux conseils pour réussir en tant que chercheur.

Fabien, merci pour tes relectures exceptionnelles, toujours précises et qui ont grandement amélioré mes articles. Merci également pour tes questions, mais surtout pour tes réponses et tes idées dès qu'il s'agissait de passer de la théorie à la pratique et de se confronter aux données complexes que sont les EEG.

Enfin, Florian, merci pour tout. Merci pour tous ces moments en dehors de la recherche, tes conseils de carrière et ton humour. Merci pour toutes tes idées, tout en me laissant la place d'en proposer aussi. Merci d'avoir toujours trouvé du temps pour discuter et échanger sur des sujets scientifiques, administratifs ou autres. Enfin, merci pour toutes les opportunités que tu m'as offertes. Merci également à Louise et à Léo de m'avoir accueilli à Okinawa puis à Rouen et à Tom de m'avoir toléré à Saint-Malo.

I would then like to thank the reviewers of this thesis, Stéphane Bonnet and Xavier Pennec. I feel that my work is closely connected to, and builds upon, the research you have conducted and continue to pursue. I was therefore particularly grateful to have you as reviewers.

I would also like to thank the other members of my PhD committee, Gabriel Peyré, Estelle Massart and Michael Tangermann, for their interesting questions and valuable comments. I was very happy and honoured to have the opportunity to present my work in front of you.

I would also like to thank all of my collaborators: Charlotte Boucherie, Laurent Bougrain, Bastien Cavarretta, Marie-Constance Corsi, Nicolas Courty, Valérie Marissens Cueva, Océane Mille, Aimi Okabayashi, Sébastien Rimbert, Clément W. Royer, Dimitri Timoz and Tristan Venot. Doing science is nice, but doing science with other people is even better. Thank you for all the exciting and interesting discussions we have had. One of my favourite things to do is to talk about science, so I would also like to take this opportunity to thank, more broadly, everyone with whom I have had the pleasure of discussing science, whether during an informal conversation over a beer, while scribbling an equation on a piece of paper, or while working through a full proof on a blackboard (or a whiteboard at Dauphine).

The place where I have talked the most about science over these past three and a half years is the LAMSADE. I would like to thank all the members of the lab for making me feel at home whenever I came to the office. Among all the members of LAMSADE, the PhD students hold a special place in my heart.

First, I would like to thank all the PhD students who welcomed me to the lab: Ariane, Alexandre, Charles, Emma, Felix, Gil, Houria, Lucas, Manolis, Matthieu, Nicolas, Pierre, Sarra, Sébastien, Sofia, Théo and Virginia. You made me feel so welcome when I arrived, and that is what made me want to come to the office every day.

Then, I would like to thank all the PhD students who joined at the same time as me or after me: Ahmad, Bochra, Camille, Christos, Henrik, Jiaxin, Alex, Bastien, Deo, Louis, Mattheo, Nour, Vasilis, Alexandre, Edouard, Emmarius, Kirti, Lorenzo, Silas and Thomas.

A special thank you to the C602 (now B139) office, also known as *Bureau de cool*. I was so happy to see you every day and to share my joys when a paper was accepted, a proof was completed or a piece of code was running without any errors. But also to share my desperation and sorrow when things did not work as expected, a class did not go as planned, or when AI was replacing us. I will keep in mind all the jokes, the *Petite pause ?* at 4 p.m., the *muffins du Crous* we shared, the LinkedIn games, and all the small moments of joy shared during the grey Parisian winters.

Un immense merci également à Marie-Clotilde, la maman des doctorants, à Christian, Meriem et Stéphanie pour leur travail essentiel au bon déroulement de la recherche au LAMSADE. Merci beaucoup à Sandra Hague pour les nombreuses missions du côté de l'INSA Rouen. Merci à Jérôme et à Joyce de prendre le temps d'écouter les doctorants, en particulier lors de cet été 2026 mouvementé marqué par le déménagement.

Pendant ces trois années de thèse, j'ai pu profiter des nombreuses boulangeries parisiennes, et j'aimerais remercier particulièrement *Éveil*, la meilleure boulangerie de Paris selon moi, mais aussi toutes celles qui m'ont permis de déguster de bons flans. Pour son soutien quasi quotidien, j'aimerais également remercier le Crous de Dauphine, malgré le peu de diversité des options végétariennes, mais qui se rattrapait quasiment toujours sur les desserts.

Avant d'arriver au LAMSADE, j'ai eu la chance de croiser sur ma route de nombreuses personnes formidables, et je suis reconnaissant d'avoir autant d'amis incroyables.

Lors de ma préparation à l'agrégation, j'ai rencontré une bande de jeunes mathématiciens aussi talentueux que sympathiques, qui ont accepté un ingénieur parmi eux et que j'ai maintenant la chance d'appeler mes amis : Bastien[1], Chloë, Mario, Roméo, Sébastian et Thaddeus. Vous ne pouvez pas imaginer à quel point votre aide durant cette année m'a été précieuse, ni combien vous avoir dans ma vie me remplit de joie. Une mention très spéciale à Mario, sans qui les théorèmes 3.18 et 3.42, ainsi que la preuve de la proposition 2.22, n'auraient pas vu le jour.

Mes années à l'ENSTA, bien qu'émaillées par les différents confinements, m'ont également permis de rencontrer un nombre impressionnant de personnes formidables. J'espère n'oublier personne en remerciant Adrien, Aimi, Amalia, Gabriel, Grégoire, Guillaume, Johan, Juliette, Laetitia, Léonard, Manon, Marin, Marion, Martin, Matthéo, Nathan, Oscar, Philippine, Raphaël, Romain, Théo, Sophie et Valentine. Vous avez tous contribué à rendre ce plateau, d'ordinaire si pluvieux et gris, un peu plus ensoleillé et coloré. J'apprécie encore beaucoup que nous réussissions à nous retrouver tous ensemble aux quatre coins de la France. Parmi les ENSTA, Valentine tient une place très particulière, et je te remercie pour tout.

Ma découverte et mon amour pour les mathématiques ont commencé à Saint-Étienne, en prépa, entouré d'une joyeuse bande de préparationnaires avec qui nous avons traversé vents et marées. Je remercie pour ces deux années Anna, Bastien, Charlotte, Francis, François, Geoffroy, Jeremy, Marwan et Valentin. J'ai aussi une pensée amicale pour Monsieur Berry, qui s'acharnait à parler de littérature et d'amour à de jeunes scientifiques peu attentifs.

Beaucoup de choses ont bien évidemment commencé au lycée René Descartes, lieu où s'est formée une équipe toujours solidaire, bien qu'un peu éparpillée. Pour ces trois années incroyables et pour toutes celles qui ont suivi, je tiens à remercier Camille, Emilien, Guillaume, Hugo, Lucas, Olivier et Pierre. Hugo et Camille, j'espère vous recroiser sur des sentiers de

[1] Troisième mention. Gourmand.

trail. Cette expérience nocturne de fin novembre reste pour moi un souvenir marquant de ces trois dernières années.

Enfin, j'ai la chance d'avoir une grande famille aimante et attentionnée, avec de nombreux cousins avec lesquels il est toujours agréable de partager un apéro, un repas ou un week-end. J'ai une pensée particulière pour mes couzinzins parisiens (ou anciennement parisiens), Anna, Constantin, Juliette et Léa.

Je ne peux terminer ces remerciements sans remercier ma sœur Marine et mes parents, qui ont toujours cru en moi, dans les moments de réussite comme dans les moments de doute. Merci de m'avoir donné les meilleures conditions possibles pour étudier et me réaliser pleinement et librement.

Contents

1 Introduction **1**
- 1.1 Learning on structured data . . . 1
- 1.2 Problem statement . . . 6
- 1.3 Overview of the thesis and the contributions . . . 7
- 1.4 Reading guide . . . 11

2 Background **15**
- 2.1 Brain-Computer Interfaces . . . 16
- 2.2 Mathematical background . . . 30
- 2.3 Related work . . . 46

I Theory **61**

3 The wrapped Gaussian distribution **63**
- 3.1 Introduction . . . 63
- 3.2 The wrapped Gaussian on the manifold of SPD matrices . . . 66
- 3.3 Estimation of the parameters of a wrapped Gaussian . . . 82
- 3.4 Extension to other manifolds . . . 86
- 3.5 Conclusion . . . 91

4 Building a probabilistic framework for ML on $\mathcal{P}_d$ **93**
- 4.1 Introduction . . . 93
- 4.2 The two Gaussians of interest . . . 95
- 4.3 Classification under the Gaussian lens . . . 96
- 4.4 Outlier detection *à la* Gaussian . . . 104
- 4.5 Kernels with a Gaussian flavor . . . 106
- 4.6 Conclusion . . . 109

5 Wrapped Flow Matching on $\mathcal{P}_d$ **111**
- 5.1 Introduction . . . 111
- 5.2 Background on Conditional Flow Matching . . . 114
- 5.3 A global framework unifying CFM on $\mathcal{P}_d$ . . . 117
- 5.4 Wrapped Conditional Flow Matching . . . 120
- 5.5 Synthetic experiments . . . 122
- 5.6 Real-world data experiments . . . 125
- 5.7 Conclusion . . . 126

Transition: from theory to applications **129**

II Application to Brain-Computer Interfaces **131**

6 One-Class Classifiers for Anesthetic State Detection **133**
6.1 Introduction . . . 133
6.2 Anesthetic states and Median Nerve Stimulation . . . 135
6.3 One-Class Classifiers . . . 137
6.4 Main Results . . . 139
6.5 Ablation study . . . 140
6.6 Discussion . . . 142
6.7 Conclusion . . . 145

7 Interpretability of Riemannian tools **147**
7.1 Introduction . . . 147
7.2 The Riemannian t-SNE . . . 151
7.3 Contribution of each electrode . . . 164
7.4 Extension to other interpretability methods . . . 169
7.5 Conclusion . . . 171

8 Discussion & perspectives **175**
8.1 Summary of the thesis . . . 175
8.2 Discussion . . . 178
8.3 Limitations . . . 181
8.4 Perspectives . . . 182
8.5 Concluding Remarks . . . 188

Bibliography **189**

A Proofs **225**
A.1 Proofs of Chapter 2 . . . 225
A.2 Proofs of Chapter 3 . . . 227
A.3 Proofs of Chapter 5 . . . 230
A.4 Proofs of Chapter 7 . . . 231

B Appendix to Chapter 5 **235**
B.1 More details on the synthetic experiments . . . 235
B.2 Details on the training and sampling times . . . 237
B.3 More details on real-world experiments . . . 238

C Appendix to Chapter 7 **239**
C.1 Effect of the perplexity on the Riemannian t-SNE . . . 239
C.2 More plots on the influence of the sensors . . . 240

D Résumé détaillé en français **243**
D.1 Introduction . . . 243
D.2 État de l'art . . . 245
D.3 La gaussienne enveloppée . . . 246
D.4 Construire un cadre probabiliste pour l'apprentissage automatique sur $\mathcal{P}_d$ . . . 250
D.5 Flots génératifs enveloppés . . . 251
D.6 Classification à une classe pour la détection d'état d'anesthésie . . . 254
D.7 Interprétabilité d'outils riemanniens . . . 257
D.8 Discussion et perspectives . . . 261

1

Chapter 1

Introduction

Contents

1.1 Learning on structured data 1
1.2 Problem statement 6
1.3 Overview of the thesis and the contributions 7
1.4 Reading guide 11

1.1 Learning on structured data

A large class of machine learning (ML) algorithms relies on quantifying similarities between data points. The underlying assumption is that nearby or similar points tend to share common properties. Therefore, given a new observation, identifying its neighbors provides information that can be exploited to infer its properties, whether by assigning it to a class in a classification task or by predicting a continuous target value in a regression setting.

Old-school algorithms such as k-nearest neighbors (Cover and Hart 1967) or kernel methods (Hofmann et al. 2008) directly rely on the assumption that nearby data points are likely to share similar properties and have proven successful in a wide range of applications. More recently, this principle has also become central to many modern machine learning methods, though in more sophisticated forms. For instance, representation learning methods such as *Contrastive Language–Image Pre-training* (CLIP, Radford et al. 2021) and *Joint Embedding Predictive Architectures* (JEPA, LeCun 2022; Assran et al. 2023) learn latent spaces in which semantically related samples are mapped to nearby representations. Similarly, a *Graph Neural Networks* (GNN, J. Zhou et al. 2020) applied to a single graph (in order to do edge prediction for example), exploits local neighborhood information (the neighboring edges) by iteratively aggregating features from neighboring nodes, while score-based and diffusion generative models (Ho et al. 2020; Yang Song et al. 2021; Lipman, R. T. Q. Chen, et al. 2023) learn smooth vector fields that vary consistently over the data manifold.

Quantifying similarities between data points can be framed as a geometry problem related to measuring a distance. Then, one might wish to learn how a point cloud is geometrically arranged in space, how to characterize this arrangement, which arrangement is more important than another and how they are related to one another. Until the 19th century, the only geometry mathematicians were aware of was the Euclidean geometry, characterized by flat planes and straight lines. Everything was based on the only locally flat approximation of the Earth humans could apprehend at the time.

Then, to measure the distance between two points, measuring the length of the straight line was the only way to go. However, taking a step back and looking at the Earth for what it is — the three-dimensional sphere $\mathcal{S}^2$ — this straight line would lead to some issues. In fact, if two points are too far apart on the sphere $\mathcal{S}^2$, the straight line would go through the Earth, which is not physically realistic. This phenomenon is illustrated in Figure 1.1. This physical constraint adds to the data what will be central in this thesis: **structure**. When the data is highly structured, using the simple Euclidean ambient geometry may lead to unfeasible points or suboptimal results. Another example from M. Berger (2003) and inspired by classical mechanics is the motion of a two-dimensional rigid body. It has three parameters: two for the location of the center of gravity and one describing its rotation around this center. Rotations are usually described by rotation matrices and one could wonder, as we did for the sphere, how to smoothly interpolate between two rotations. This is an interesting question in mechanics, as it tells us how to rotate an object from one position to another. We illustrate this phenomenon in Figure 1.2. We can see that using the Euclidean geometry will distort the rigid body (it will shrink in this example), which is not possible in a real physical world. Indeed, rotation matrices are highly structured, they belong to the set SO(2) of special orthogonal matrices, and one must use a specific geometry to keep consistent results. As with the previous examples, graphs are also very structured: each node has edges connecting it to other nodes. As for the sphere example above, to navigate from one node to another, one must respect the topology of the graph, by following the edges. In the case of graphs, there is no "classical geometry" inherited from the ambient space that can be used and in most applications, decisions must be invariant to permutations (ie. to the ordering of the nodes).

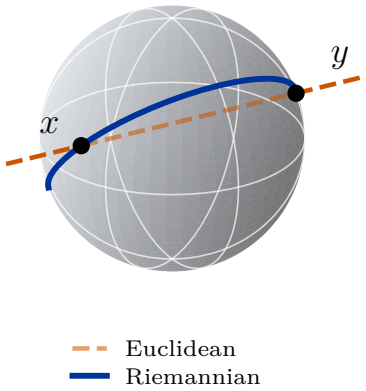


Figure 1.1. The sphere $\mathcal{S}^2$. Its curved geometry needs to be taken into account to build a coherent path between two points x and y on the sphere.

In order to deal with these types of objects, mathematicians had to free themselves from the Euclidean geometry only to discover that this Euclidean geometry is one special type of geometry that co-exists among others. During the 19th century, mathematicians like Gauss, Bolyai or Lobachevsky discovered the hyperbolic geometry (Jenkovszky et al. 2023), then Riemann (1867) formalized the notion of what is now called *Riemannian manifolds* and that will be central in this thesis. This geometry encompasses the Euclidean geometry, while allowing for other geometries to exist. This is in particular the solution to the problem we had when interpolating between points on the sphere or on SO(2). These two spaces are instances of Riemannian manifolds, roughly meaning that locally, they look flat like a Euclidean space, but globally, they can be curved. This Riemannian geometry gives us the tools to measure distances or angles between points, continuously interpolating between points, computing trajectories and measuring their speed and direction, and even characterizing the local curvature of a curved space. For example, the Euclidean geometry is flat, so its curvature is 0, while the curvature of the sphere is 1. Intuitively, curvature quantifies how a multidimensional space deviates from a flat Euclidean space, with its effects manifesting locally in the behavior of vectors, geodesics, and volumes. In particular, positive sectional curvature tends to make initially parallel geodesics converge, whereas negative sectional curvature tends to make them diverge.

While mathematicians have well understood the importance of choosing the correct geometry for structured data, computer scientists have based almost all of their algorithms

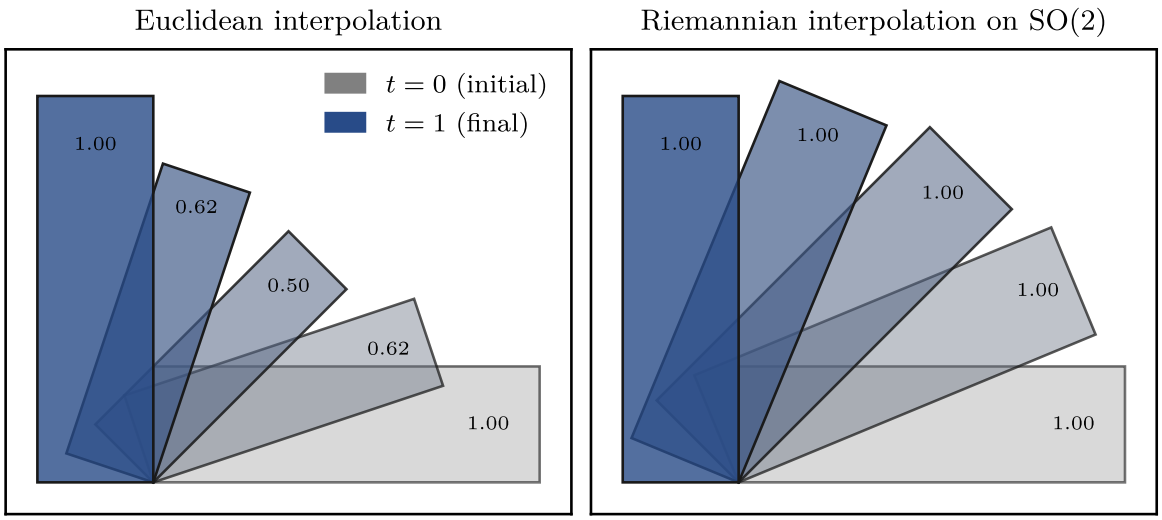


Figure 1.2. Illustration of the interpolation of a rigid body (here a rectangle) between two positions. The number in the rectangle corresponds to its area. On the left panel, the interpolation is done using the Euclidean geometry of SO(2), the rigid body gets distorted, which is not physically realistic. On the right panel, the Riemannian geometry of SO(2) is taken into account, and no distortion is visible.

on the assumption that the data was Euclidean, thus relying on the flat Euclidean geometry. As noted by Papillon et al. (2025), it has only been two decades since the machine learning field has expanded to new horizons, moving beyond flat Euclidean spaces. A clear example is the rise of *Graph Neural Networks* (GNN) for dealing with graph data (Bronstein, Bruna, Cohen, et al. 2021). The key design element of GNN is the use of *pairwise message passing*, an operation that allows each node to update its own representation by aggregating information (features) from its immediate neighborhood. This operation directly reflects the geometry of the graph: close-by nodes will communicate more than nodes that are very far away. GNNs have proven to be very efficient in antibiotic discovery (Stokes et al. 2020), in solving the protein folding problem (Jumper et al. 2021), in traffic predictions (Derrow-Pinion et al. 2021), or designing web-scale recommender systems (R. Ying et al. 2018).

The main motivation: Brain-Computer Interfaces

In this thesis, the main motivation and application is *Brain-Computer Interface* (BCI). A BCI is a system that is able to translate the user's brain activity into real-life commands for an interactive application (Clerc et al. 2016). The first step for achieving this goal is to record the electrical activity of the brain of a subject. Here, we focus on *ElectroEncephaloGraphy* based BCI: a set of electrodes is placed on the scalp of the subject, each recording the electrical activity of a certain part of the brain, forming a multivariate time-series called an *ElectroEncephaloGram* (EEG). The goal of the BCI is to decode this EEG into a command, in order to move a prosthesis, command a wheelchair or type on a virtual keyboard for example. A high-level BCI pipeline is illustrated in Figure 1.3, and we will give more details on how a BCI works in Section 2.1. Classical machine learning tools used in Motor Imagery BCI (see Section 2.1) were traditionally trained on the extracted power of the different electrodes in specific frequency bands (see Chapter 7 of Clerc et al. 2016). This power information can be retrieved by computing the variance of the signal of the different electrodes. Modern BCI

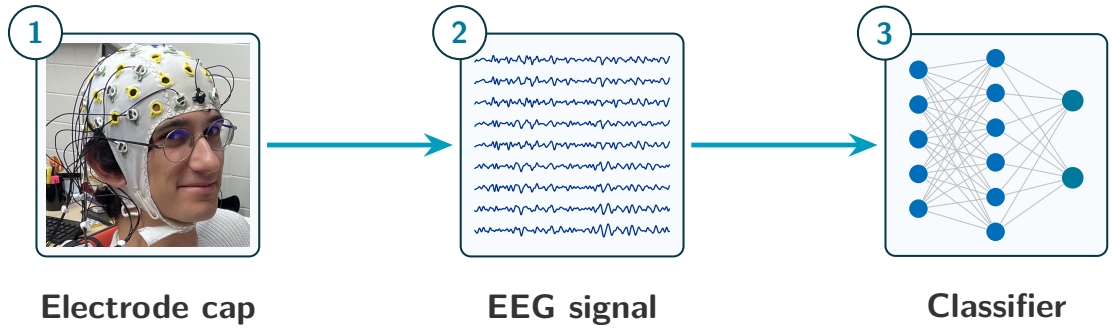


Figure 1.3. The typical pipeline of an EEG-based Brain–Computer Interface: electrical activity is picked up by an electrode cap while the subject performs a mental action, such as motor imagery. This yields a multichannel EEG signal that a classifier turns into a command.

pipelines do not only take into account the variance of the signals, but also their covariances (Congedo et al. 2017; Yger, Berar, et al. 2017). All of this information is summarized in the spatial *covariance matrix* of the EEG. Thus, the main feature we will study in this thesis is the covariance matrix of a multivariate time-series. Covariance matrices will be estimated from the EEGs, and the input of the learning tools studied will always be a point cloud of covariance matrices.

Like points on the sphere, rotation matrices and graphs, covariance matrices are highly structured objects. Indeed, under some assumptions, they belong to a specific set of matrices: the set of *Symmetric Positive Definite* (SPD) matrices that we denote $\mathcal{P}_d$. Here, the dimension d of the studied matrices corresponds to the number of electrodes considered. Here, once again, using the flat Euclidean geometry on SPD matrices can be problematic, but in a slightly different way than on $\mathcal{S}^2$ or $\mathrm{SO}(2)$. Indeed, when interpolating between two SPD matrices, the set being convex, the interpolation will always be SPD. However, extrapolating is not possible, and when interpolating between two SPD matrices of the same volume, the interpolation will distort the volume. This notion of volume and the effect known as *swelling effect* (Arsigny et al. 2006a) will be detailed in Section 2.2.2. Thus, a curved Riemannian geometry is more appropriate when dealing with SPD matrices (Yger, Berar, et al. 2017; Pennec 2020). SPD matrices are not only present in BCI, they are also used in medical imaging, for *Diffusion Tensor Magnetic Resonance Imaging* (DT-MRI, or simply DTI) (Le Bihan et al. 2001) where the DTI are SPD-matrix-valued images. SPD matrices have been used as covariance descriptors in computer vision (Tuzel et al. 2006) or in process control (Iruthayarajan and Baskar 2010). They also appear in robotics (Jaquier and Rozo 2027), where they represent the inertia matrix of multi-body systems (Friedl, Jaquier, Lundell, et al. 2025; Friedl, Jaquier, Liao, et al. 2025) or the control gain matrices (Marco et al. 2016).

Even if we are capable of building BCIs that work, the field still faces some challenges. One of the main ones is that the recorded EEGs are subject to numerous variabilities (Fairclough and Lotte 2020; Grosse-Wentrup and Schölkopf 2013). Indeed, an EEG recorded while the subject performs the same mental task will differ based on the time of day, the stress or mental workload of the subject, their environment or other unknown variables. This is a real problem preventing the development of robust BCIs beyond academic laboratories. Another issue is that collecting EEG data is very expensive (Lotte, Dreyer, et al. 2025), thus making it difficult to rely on deep learning methods that require a huge amount of training data. In this context, relying deeply on the structure of the data, here of SPD matrices, can be a solution to alleviate data scarcity.

Incorporating structure through equivariance and invariance

Taking into account the structure of the data can also help to simplify the problem. In particular, core ideas in geometric deep learning (Bronstein, Bruna, LeCun, et al. 2017) are that of *equivariance to symmetry groups* and *invariance to symmetry groups* (S.-i. Amari 1978; Makadia et al. 2007; Cohen and Welling 2016; Bruna and Mallat 2013; Zaheer et al. 2017). A network (or a layer of a network) f is said to be *equivariant* to a symmetry group G if $f(g \cdot x) = g \cdot f(x)$ for all elements $g \in G$ and all inputs x, and f is said to be *invariant* to G if $f(g \cdot x) = f(x)$. Thus, an equivariant network gives the same result when first transforming the input and then applying f as first applying f and then transforming the output while an invariant network's output does not change at all under the transformation.

Famous equivariant and invariant layers are *Convolutional Neural Networks* (CNN) that are equivariant to translation: if L_t shifts an image $\boldsymbol{X}$ by t, a convolutional layer Conv satisfies $\text{Conv}(L_t\boldsymbol{X}) = L_t\text{Conv}(\boldsymbol{X})$: shifting the input shifts the feature map by the same amount. Adding global average pooling on top makes the whole network translation-invariant: an object is recognized regardless of where it sits in the image. In Cohen and Welling (2016), the authors generalized convolutions to be equivariant to rotations, so that feature maps rotate consistently with the input. For a graph, permuting node order should permute node features the same way, thus being equivariant to node permutations. Message-passing GNNs are permutation-equivariant at the node level (Bronstein, Bruna, Cohen, et al. 2021). On the contrary, a downstream sum, mean or max over all nodes for graph classification is permutation-invariant (Zaheer et al. 2017). When dealing with SPD matrices, the core mathematical object of this thesis, we will see in Section 2.2 that we can consider a Riemannian metric that is *affine-invariant*, thus invariant under the action of congruence by an invertible matrix: $\boldsymbol{X} \mapsto \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^\top$, where $\boldsymbol{X}$ is SPD and $\boldsymbol{A}$ is invertible. Equivariant and invariant networks have improved the performances and robustness in very diverse fields as quantum mechanics (Gerdes et al. 2023), generative models of molecules (Igashov et al. 2024), particle physics (Spinner et al. 2024) and robotics (Brehmer, Bose, et al. 2023).

The core benefit of using equivariant/invariant layers is that the network does not have to learn the geometry from the data. Without them, a network has to discover invariance by looking at enough examples of different transformed versions of the data. An equivariant/invariant layer makes the symmetry exact by construction, for every input, including ones never seen during training. This will eventually help the network to generalize to new unseen transformations. Moreover, it will reduce the effect of certain types of variabilities. For example, an invariant network does not care about the orientation of an image, or about the order of the nodes. In BCI, we will see in Section 2.1.4 that the *affine-invariant* discussed above removes effects of electrode repositioning or changes in the head geometry.

One could argue that with modern deep learning methods and a sufficient amount of data, one could learn the structure of the data without encoding it *a priori*. If the network is sufficiently deep and well-trained, maybe a non-equivariant or non-invariant network would lead to the same performances as equivariant/invariant ones. In Brehmer, Behrends, et al. (2025), the authors tested this hypothesis on a benchmark problem of rigid-body interactions. Rigid bodies are invariant under the group E(3) of isometries of Euclidean space that consists of translations, rotations, and reflections. They showed that invariant transformers are more data-efficient, but that data augmentation could close this gap. If provided with a sufficient amount of data and trained long enough, non-invariant architectures are indeed able to learn symmetries from data. However, they also show that for a fixed compute budget, invariant networks outperform non-invariant ones.

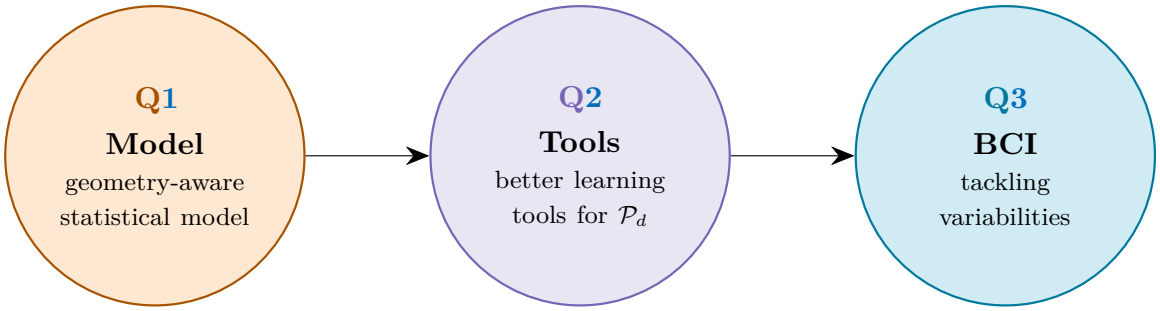


Figure 1.4. The three questions of this thesis build on one another: question 1 sets up a geometry-aware model, question 2 turns it into learning tools, and question 3 returns to the BCI application, where the model itself is used directly and not only through the tools built from it.

1.2 Problem statement

In modern machine learning, the core assumption is that **data itself comes from a probability distribution** (Vapnik 2013; Hastie et al. 2009). Everything can then be described in terms of probability distributions: the set of realistic images? It is a distribution over the space of pixel values $\mathbb{R}^{H\times W\times 3}$. The set of English words? It is a distribution over the possible chains of characters given the Latin alphabet. The set of actions a robot takes in order to perform a certain task? It is a distribution over actions conditioned on the current state. Then, surely, the set of covariance matrices representing realistic EEGs is a probability distribution over the set of covariance matrices.

This paradigm will be at the heart of this thesis. However, we saw that taking into account the structure of the data was essential in order to build coherent models. In this thesis, we focus on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices, thus, relying on Euclidean probability distributions does not make sense, as it does not properly take into account the curved geometry of SPD matrices. The main question we ask in this work is:

> *Question 1.* What geometry-aware statistical models yield useful representations on the manifold of SPD matrices?

In order to solve this question, we will develop a geometry-aware version of the multivariate Gaussian distribution for the manifold of SPD matrices, that is called the *wrapped Gaussian*. Of course, the study of probability distributions on manifolds is not a new topic, and we will argue why this wrapped Gaussian best fits our needs.

Once we can model data on the manifold of SPD matrices using a robust probability distribution, we can transform it into a learning method. Indeed, using probability distributions to model the data is only the first step of building a learning method. This naturally brings up a second question:

> *Question 2.* Can this statistical model help design better learning tools for SPD matrices?

We will see that using different Gaussians, we are able to build a probabilistic framework for learning tools used on $\mathcal{P}_d$ and also to propose extensions of known tools using the wrapped Gaussian.

Finally, recall that our main application in this thesis is Brain-Computer Interfaces (BCI), and more precisely the Riemannian framework of BCI that considers the covariance matrices as the main feature. After asking theoretical and algorithmic questions, we are interested in going back to our application. Thus, the third question addressed in this manuscript is:

> *Question 3.* Can one use this statistical model to better tackle variabilities observed in BCI?

We will focus on a specific application, that is detecting whether a patient is asleep or

awake during general anesthesia. We will show that using our wrapped Gaussian, we are able to build a better one-class classifier, and we argue that this is due to a better handling of variabilities.

1.3 Overview of the thesis and the contributions

This thesis is divided into two parts. The first part is more theoretical, and aims at answering question 1 by building a *wrapped Gaussian* and using it to put together a probabilistic framework for learning tools on the manifold of SPD matrices, giving a first answer to question 2. In the second part, we focus more precisely on the application of BCI by first, focusing on a specific type of BCI used to detect consciousness during general anesthesia. This will give a second answer to question 2 as well as to question 3. Finally, we will open up some interpretability concerns, by building tools to better apprehend the decision taken by Riemannian BCI pipelines.

We will start with Chapter 2 by giving a complete overview of BCIs, how they work, some interesting applications and some of its challenges. Then, we will motivate the use of the covariance matrix as a feature for BCI and give the mathematical background on SPD matrices needed to understand the rest of this manuscript. For this, after recalling some properties of SPD matrices, we will justify the problems caused by the Euclidean geometry and describe several different other geometries one can use for this set. Finally, we will describe thoroughly existing learning methods on Riemannian manifolds, not only focusing on the manifold of SPD matrices. The different methods reviewed will be classified into two categories: *intrinsic* methods that are defined directly on the manifold, and *extrinsic* methods that rely on a local linearization of the manifold. This distinction will be important throughout the manuscript.

Part I: Theory

Chapter 3: The wrapped Gaussian distribution In order to answer the main question 1, we decided to extend the multivariate Gaussian to the manifold $\mathcal{P}_d$ of SPD matrices. In order to do so, we leverage a Euclidean linearization, the tangent space, and rely on the Riemannian exponential mapping to send the Euclidean multivariate Gaussian from the tangent space to the manifold. This is called a *wrapped distribution*. Compared to other Gaussian distributions defined on Riemannian manifolds, this wrapped distribution has some advantages such as its simple sampling algorithm, its closed-form density and most importantly: its anisotropy. It is precisely this anisotropy that will make the wrapped Gaussian useful in practice. We will also encounter an identifiability issue, a counterpart of a very expressive statistical model. This issue will be solved by considering equivalence classes of parameters. Then, the core problem of parameter estimation for the wrapped Gaussian will be tackled, and a solution based on Riemannian optimization will be proposed. Finally, we will see how this wrapped Gaussian can be extended to other manifolds.

This chapter is based on the following paper published at ICML 2025:

> **Thibault de Surrel**, Fabien Lotte, Sylvain Chevallier, and Florian Yger (2025). "Wrapped Gaussian on the manifold of Symmetric Positive Definite Matrices". In: *Forty-second International Conference on Machine Learning*

The version included in this thesis improves upon the ICML version: it provides stronger theoretical results, a complete characterization of identifiability for wrapped Gaussian distributions on $\mathcal{P}_d$, and an overall clearer presentation. The extension of the wrapped Gaussian to Hadamard manifolds discussed in Section 3.4 is currently under review at the journal *Foundation of Computational Mathematics* (FoCM):

> **Thibault de Surrel**, Sylvain Chevallier, Fabien Lotte, and Florian Yger (2026). "Wrapped Gaussian on Hadamard Manifolds". Under review at *Foundations of Computational Mathematics*

Chapter 4: Building a probabilistic framework for ML on $\mathcal{P}_d$ Many Euclidean learning algorithms are based on the foundational assumption that the data is distributed based on a Gaussian. This encompasses classification algorithms, outlier detection algorithms or kernels. Some of these learning methods have been extended to the manifold $\mathcal{P}_d$, in order to deal with inputs that are SPD matrices and to respect the curved geometry. In this chapter, we revisit different learning tools on $\mathcal{P}_d$ using two different Gaussians: the isotropic Gaussian from Said, Bombrun, et al. (2017) and the wrapped Gaussian built in Chapter 3. More precisely, we show how classification methods, outlier detection algorithms as well as kernels, can be rewritten in the light of the two Gaussians mentioned previously. When they are based on the isotropic Gaussian, we propose extensions using the wrapped Gaussian, which has the advantage of being anisotropic. This helps model complex data distributions that can exhibit privileged directions or directional bias. We also conduct some numerical experiments to compare the different methods. This chapter gives a first answer to question 2, using the wrapped Gaussian to build new learning tools.

This probabilistic framework for the classifiers, along with the two newly proposed classifiers, was published in the ICML paper on wrapped Gaussians:

> **Thibault de Surrel**, Fabien Lotte, Sylvain Chevallier, and Florian Yger (2025). "Wrapped Gaussian on the manifold of Symmetric Positive Definite Matrices". In: *Forty-second International Conference on Machine Learning*

The numerical comparison of the different classifiers is also taken from this paper. The full probabilistic framework, except for the kernels, was later published in:

> **Thibault de Surrel**, Florian Yger, Fabien Lotte, and Sylvain Chevallier (2025). "A probabilistic view on Riemannian machine learning models for SPD matrices". In: *International Conference on Geometric Science of Information*. Springer, pp. 142–151

Here, we present an enhanced version of this probabilistic framework: we add an analysis of Riemannian kernels, propose a new kernel based on the wrapped Gaussian, and compare the different kernels through numerical experiments.

Chapter 5: Wrapped Flow Matching on $\mathcal{P}_d$ One of the main advantages of the Euclidean Gaussian distribution is that sampling algorithms have been deeply studied in the past 60 years, leading to efficient and reliable algorithms. This advantage also transfers to the wrapped Gaussian on $\mathcal{P}_d$ as the sampling revolves around an explicit transformation of Euclidean Gaussian samples. A specific type of generative models, named *Flow Matching*, relies on the ease of sampling a Gaussian to create new samples from an unknown data distribution from which we only have access to a finite number of samples. In this chapter, using the wrapped Gaussian from Chapter 3, we show how this Flow Matching model can be extended to the manifold of SPD matrices. Moreover, we are able to extend the probabilistic framework built in Chapter 4 to those Flow Matching models. Indeed, we rewrite state-of-the-art Riemannian Flow Matching models from Z. Chen, Yue Song, G. Liu, et al. (2024) and Collas, Ju, et al. (2026) using a wrapped Gaussian and propose a new version of this algorithm, offering theoretical advantages compared to the two state-of-the-art methods. This work gives another answer to question 2, as the wrapped Gaussian is, once again, the underlying link between the different Flow Matching methods studied.

This work was done in collaboration with a PhD student of IRISA, Aimi Okabayashi, and is published in the journal *Transactions on Machine Learning Research*:

> Thibault de Surrel, Aimi Okabayashi, and Nicolas Courty (2026). "Wrapped Flow Matching for Symmetric Positive Definite Matrices". In: *Transactions on Machine Learning Research*

∗ These two authors contributed equally to this work.

In this manuscript, we propose new illustrations to better understand Flow Matching as well as the differences between the different geometry-aware Flow Matching methods that are studied.

Part II: Application to Brain-Computer Interfaces

Chapter 6: One-Class Classifiers for Anesthetic State Detection Our main motivation for building a probabilistic model on the manifold of SPD matrices is to better handle variabilities in BCI. In this chapter, we tackle a specific problem of detecting consciousness during general anesthesia. Indeed, during general anesthesia, some patients can experience what is called an *Accidental Awareness during General Anesthesia* (AAGA), an unintended episode of consciousness. This experience can be traumatic for the patient and a first step in order to detect AAGA is to build a classifier capable of detecting whether a patient is awake or anesthetized. Thanks to my co-authors for this work, we got access to a dataset of EEGs recorded before and during general anesthesia. Using the wrapped Gaussian built in Chapter 3 and the methods developed in Chapter 4, we propose a new anisotropic One-Class Classifier for SPD matrices called *One-Class WG*. This One-Class Classifier is trained only on the awake state and its goal is to detect deviations from this baseline state, that, in our case, will correspond to the anesthetized state. We show that the proposed One-Class WG outperforms SOTA methods and analyze the results to better understand the reasons of this improvement. This will help us answer question 3.

This work was done in collaboration with a PhD student at INRIA Bordeaux, Valérie Marissens Cueva and led to a publication at the Graz BCI Conference:

> Valérie Marissens Cueva*, **Thibault de Surrel***, Laurent Bougrain, Seyed Javad Bidgoli, Guy M Cheron, Ana Maria Cebolla Alvarez, Claude Meistelman, Fabien Lotte, Florian Yger, and Sébastien Rimbert (2026). "Wrapped One-Class Riemannian EEG Classifier for BCI-Based Detection of Anesthetic States". In: *10th International Graz Brain-Computer Interface Conference*

∗ These two authors contributed equally to this work.

Chapter 7: Interpretability of Riemannian tools Building a model performing well is good, but understanding why it performs well is better. This is the goal of interpretability. In this final chapter, we take a step aside from the problem of building a better Riemannian classifier for BCI to try and tackle the interpretability question: why does this model make this decision when given this input? This is an important question for all BCI stakeholders, from the BCI researcher building the interface, the doctor taking decisions based on the output of the BCI to the final user using the BCI. The decision must be understood by the different actors at play, and the better they understand the decision, the more trust they will put in Riemannian BCIs. In order to answer this question, we start by developing a visualization tool for high-dimensional SPD matrices, by adapting the well-known t-SNE algorithm from Maaten and Hinton (2008) to the Riemannian manifold of SPD matrices. Then, we also study different interpretability tools, adapt them to work with SPD matrices, and show that the

results of a Riemannian classification pipeline in BCI are coherent with neurophysiological knowledge.

The first part of Chapter 7 on the Riemannian t-SNE builds on the following paper, published in the journal *Transactions on Machine Learning Research*:

> **Thibault de Surrel**, Sylvain Chevallier, Fabien Lotte, and Florian Yger (2025). "Geometry-Aware visualization of high dimensional Symmetric Positive Definite matrices". In: *Transactions on Machine Learning Research*

Compared to the published version, we present here an improved algorithm, in particular from the numerical standpoint, achieving a significant speed-up thanks to mathematical and optimization tricks.

The second part on the influence of each electrode draws on the following paper, published at MLSP 2025:

> **Thibault de Surrel**, Tristan Venot, Marie-Constance Corsi, and Florian Yger (2025). "Interpretability of Riemannian tools used in Brain Computer Interfaces". In: *2025 IEEE 35th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, pp. 1–6

Here too, we offer a fresh take on this work, introducing new methods and extensions, developed in part with the help of our intern at LITIS, Océane Mille.

Other works

This work on Riemannian BCIs started as an M2 internship in April 2023. During this internship, we started by developing a new classifier based on the *Minimum Distance to the Mean* (MDM) classifier from Barachant, Bonnet, et al. (2011). Instead of estimating a single covariance matrix per EEG trial, we cut the EEG into several smaller windows and estimated a covariance matrix per time window. This led to a *trajectory of covariance matrices*, instead of a unique covariance matrix per EEG trial. Then, we computed the mean trajectory of each class, based on two different methods, and classified a new trajectory based on the Riemannian distance or the *Dynamic Time Warping* distance from Berndt and Clifford (1994) to the different means. This led to a publication at EUSIPCO 2024 that is not discussed in this manuscript:

> **Thibault de Surrel**, Florian Yger, Sylvain Chevallier, and Fabien Lotte (2024). "Averaging trajectories on the manifold of symmetric positive definite matrices". In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1182–1186

During my PhD, I also worked on other topics that are not included in this manuscript. In particular, I also studied a deep learning architecture built for SPD matrices: the *SPDNet* from Huang and Van Gool (2017) that will be described in Section 2.3.5. I had the chance to help Florian Yger supervise two interns on this topic. The first one was Charlotte Boucherie that worked on building an SPDNet autoencoder specifically designed to encode high-dimensional SPD matrices into low-dimensional ones while still being able to reconstruct them correctly. The work led to a publication at ESANN 2026:

> Charlotte Boucherie*, **Thibault de Surrel***, and Florian Yger (2026). "SPDNet-AE: a Compact SPD Representation through Riemannian Autoencoding". In: *ESANN*

∗ These two authors contributed equally to this work.

The second intern, Dimitri Timoz, focused on adversarial attacks on the SPDNet. We were the first ones to attack this network, and a first step was to work in a white-box scenario, in which one knows how to compute the gradient of the loss. In this context, we adapted the *Projected Gradient Descent* algorithm to the manifold of SPD matrices in order to find a perturbation that fools the trained SPDNet. This work led to a publication at ICASSP 2026:

> Dimitri Timoz, **Thibault de Surrel**, and Florian Yger (2026). "Riemannian adversarial attacks on Symmetric Positive Definite matrices". In: *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5356–5360

This first work on adversarial attacks on SPDNet was in a *white-box* setting: we assumed that we knew the weights of the model, thus allowing for the computation of gradients. This scenario is not very realistic, thus, with another PhD student at LAMSADE, Bastien Cavarretta, we extended this setup to *black-box attacks*, where we only have access to the outputs of the network. In order to attack the SPDNet based solely on its outputs, we used Riemannian zeroth-order algorithms such as the *Riemannian Random Search* or the *Riemannian Direct Search* based on derivative free optimization. This work is currently under review at AAAI 2027:

> Bastien Cavarretta, **Thibault de Surrel**, Clément W. Royer, and Florian Yger (2026). "Riemannian Black-box Adversarial Attacks on SPD Networks". Under review at *AAAI 2027*.

During my PhD I also participated in several *My PhD in 3 minutes* contests: one organized by the conference EUSIPCO 2024 where I won the *Audience Prize runner-up (2nd place)* (**de Surrel** 2025) and the *Ma thèse en 180 secondes* organized by PSL in 2025.

1.4 Reading guide

We now provide a short reading guide of this manuscript. In the first two chapters (including the one you are finishing reading right now), the general motivation and goals of this thesis are discussed and the necessary background on Brain-Computer Interfaces, geometry of SPD matrices and related works are given. Chapter 3 introduces the main theoretical object of this thesis: the *wrapped Gaussian*. Then, Chapter 4 and Chapter 5 can be read independently, while we recommend reading Chapter 4 before Chapter 5. Chapter 6 is the continuity of Chapter 4, rather focusing on a specific application and a specific algorithm. Finally, Chapter 7 is quite independent of the rest, it is the only chapter that does not rely on the wrapped Gaussian. We summarize this reading guide in Figure 1.5.

As described in Section 1.3, most of the work presented here was also published (or is currently under review) in conferences or journals. However, sometimes, we have added new ideas, developments or experiments in this manuscript. When this is the case, this green badge will appear in the margin. If this new result is of interest for you, please consider citing this thesis rather than the related article.



A note on reproducibility

A claim that cannot be checked is of little use, and this is especially true for empirical results on EEG data, where preprocessing choices alone can change the conclusions. I have therefore gathered (most of) the code used to produce the figures, tables and results of this manuscript in the following GitHub repository:

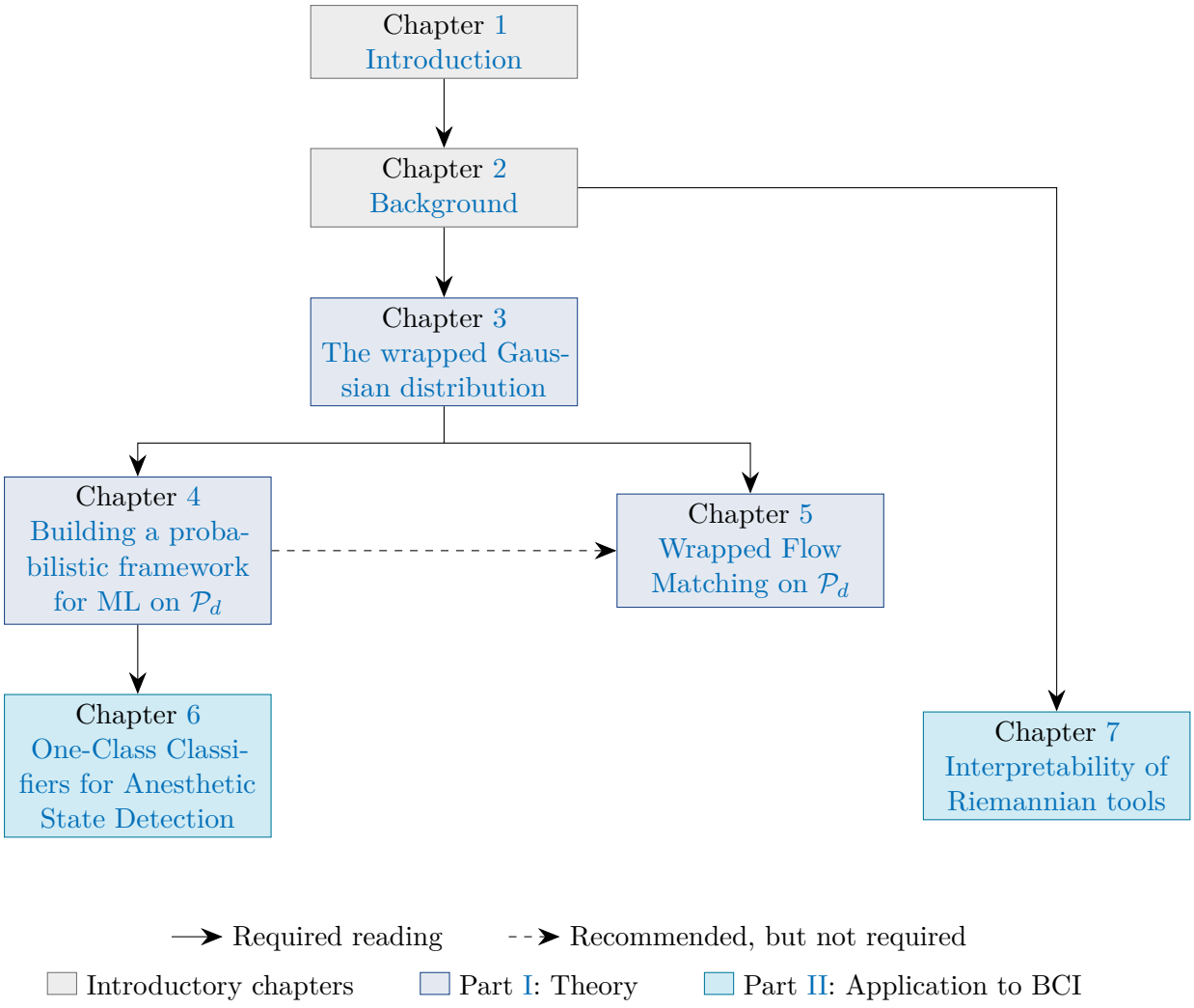


Figure 1.5. Reading guide of this thesis.

https://github.com/thibaultdesurrel/Code-for-manuscript

The repository is organised chapter by chapter, and each script is named after the figure or the table it produces, so that a specific result of this manuscript can be traced back to the code that generated it. Most of the experiments rely on publicly available datasets, loaded and preprocessed through MOABB (Aristimunha, Carrara, et al. 2026), and on the standard Riemannian and machine learning stack – mainly pyRiemann (Barachant, Barthélemy, et al. 2026) and Scikit-learn (Pedregosa et al. 2011)– so that they can be re-run end to end without any manual data collection.

The code of Chapter 5 on Flow Matching lives in a separate repository, as this project required substantially more code than the others:

https://github.com/thibaultdesurrel/Wrapped_Flow_Matching_SPD

The only exception is Chapter 6 on One-Class classifiers. The EEG signals used there were recorded during real surgeries under general anesthesia, and cannot be shared publicly for patient confidentiality reasons. The experiments of that chapter can therefore not be reproduced as such. The implementation of the One-Class wrapped Gaussian is nevertheless released with the rest of the code, and can be applied to any other dataset of SPD matrices.

Finally, beyond this manuscript, I tried to make the tools developed during this PhD directly usable by the community rather than leaving them in a repository: the Riemannian

t-SNE of Chapter 7 as well as the sampling of the wrapped Gaussian studied in Chapter 3 have, for instance, been contributed to pyRiemann (Barachant, Barthélemy, et al. 2026), and are now available to any user of the library.

2

Chapter 2

Background

Contents

2.1 Brain-Computer Interfaces 16
2.1.1 What are BCIs and how do they work? 16
2.1.2 Applications of BCIs 23
2.1.3 Challenges of BCIs 24
2.1.4 Using covariance matrices and their Riemannian geometry to build better BCIs 27
2.2 Mathematical background 30
2.2.1 Matrix properties 31
2.2.2 What geometry for SPD matrices? 32
2.2.3 A Riemannian metric for the set of SPD matrices 34
2.2.4 A vectorization of the tangent spaces 44
2.3 Related work 46
2.3.1 Supervised learning 46
2.3.2 Kernel methods 47
2.3.3 Dimension reduction 48
2.3.4 Probability distributions on manifolds 50
2.3.5 Deep learning on manifolds 56
2.3.6 Classifying the SOTA methods between intrinsic and extrinsic methods 58

In the introduction of this thesis given in Chapter 1, we framed the main questions of this thesis around machine learning and statistical models built for the Riemannian manifold $\mathcal{P}_d$ of SPD matrices, and their application to Brain-Computer Interfaces (BCI). We can therefore identify three main components to the journey described in this manuscript:

1. Brain-Computer Interfaces, which is the application studied, thus the final destination we want to reach,
2. The Riemannian manifold of SPD matrices, which is the world we will navigate in this thesis,
3. Machine learning and statistical models, which are the tools we will use to find our way during this journey.

The goal of this chapter is to provide a comprehensive introduction to each of these components, in order to give the reader the knowledge necessary before embarking on the trip proposed by the following chapters. The first section of this chapter, Section 2.1, will be focused on Brain-Computer Interfaces. We will see what they are, give an overview of

how they work and some of the applications of such a technology. More importantly, we will describe the current challenges BCIs are facing, some of which we aim at addressing in this work. Finally, we will build the bridge between BCIs and the Riemannian geometry of SPD matrices, by justifying the use of covariance matrices as a good feature for these interfaces.

Then, once we are all convinced that the covariance matrix is the right feature for BCIs, we will review the mathematics required to deal with them in Section 2.2. More generally, we will focus on SPD matrices, arguing why the Euclidean geometry is not well suited when dealing with such matrices, and we will explore the different geometries that have been defined on this set. We will also compare the different geometries between them, both from a theoretical and a practical perspective.

Finally, in Section 2.3, we present a thorough review of the literature on machine learning and statistics over manifolds, covering supervised learning, kernel methods, dimensionality reduction, probabilistic modeling on manifolds, and deep learning architectures for manifold-valued data. We will also provide a classification of state-of-the-art methods into two categories: intrinsic methods on a manifold or methods relying on a tangent space. This dichotomy will be the focus of Chapter 4 where we will see the theoretical foundations of the two categories using the wrapped Gaussian defined and studied in Chapter 3.

2.1 Brain-Computer Interfaces

The main applications studied in this thesis are *Brain-Computer Interfaces* (BCIs), whose general goal is to translate the user's brain activity into actionable commands for an interactive application, forming a closed-loop system (Millán et al. 2010; Nam et al. 2018). This goal can be divided into different sub-steps: measuring the user's brain activity, preprocessing this data, translating it into a command and finally, executing this command and providing feedback to the user. The initial target population of BCIs was people with motor disabilities who still have some cognitive functions, so that BCIs could help them control assistive devices such as prostheses, spellers or other similar devices. However, in recent years, the scope of applications of BCIs has widened, enabling their use for virtual reality purposes, entertainment, user experience monitoring and neuroadaptive technologies, rehabilitation of motor and cognitive functions, and other tools for studying human brain functions (Värbu et al. 2022; Gert Pfurtscheller et al. 2003).

In what follows, the goal is to describe more thoroughly what BCIs are, how they work and for what kind of applications they have been used. Then, we will see that BCIs are subject to various kinds of variabilities, both when considering a given subject (intra-subject variabilities), and when considering several subjects (inter-subject variabilities). Finally, we will describe a specific approach for tackling this problem through the lens of Riemannian geometry. This approach has led to robust classifiers and has become state-of-the-art in BCI competitions. This thesis positions itself within this Riemannian line of work on BCIs.

2.1.1 What are BCIs and how do they work?

Let us start by giving the official definition of a BCI, as proposed by the BCI Society in Slutzky et al. (2025):

> "A brain-computer interface is a system that measures brain activity and converts it in (nearly) real-time into functionally useful outputs to replace, restore, enhance, supplement, and/or improve the natural outputs of the brain, thereby changing the ongoing interactions between the brain and its external or internal environments. It may additionally modify brain activity using targeted delivery of stimuli to create functionally useful inputs to the brain."

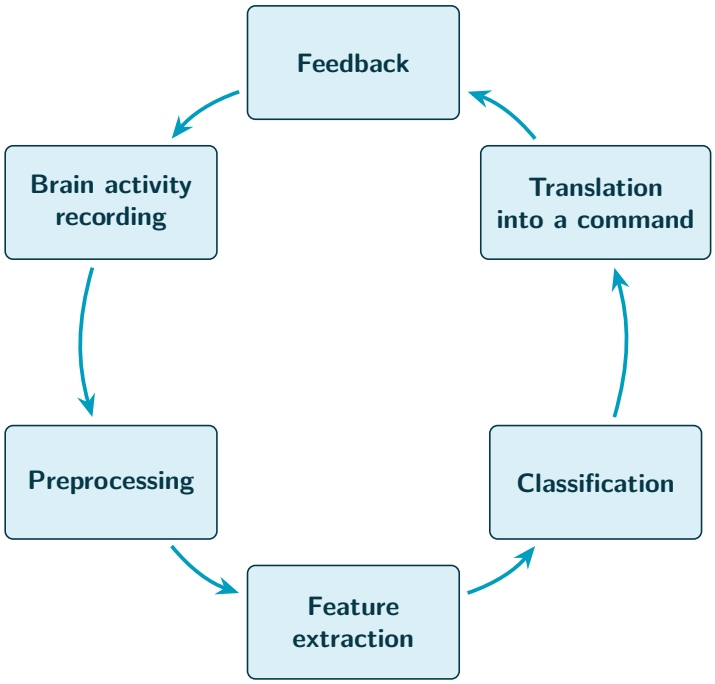


Figure 2.1. The closed-loop pipeline of a Brain–Computer Interface: the user's brain activity is recorded, preprocessed, turned into features and classified, the resulting class is translated into a command, and feedback is returned to the user, closing the loop.

This definition was agreed upon by the members of the BCI Society, an organization driven by more than 500 members whose goals are to advance BCI research, education, and collaboration around these technologies. This definition can seem abstract, and we will give concrete examples of BCIs in Section 2.1.2.

Let us now look under the hood of BCIs and see how they are built. As stated in Clerc et al. (2016), "Building a BCI is a complex and difficult task that requires knowledge of several disciplines such as computer science, electrical engineering, signal processing, neuroscience and psychology". We divide this complex system into six main stages, forming a closed loop and summarized in Figure 2.1: brain activity recording, preprocessing, feature extraction, classification, translation into a command and feedback. In some modern approaches relying on deep learning, the feature extraction and classification stages tend to be merged. Let us now describe the main steps.

Brain activity recording The goal of this first step is to acquire raw signals that reflect the user's brain activity.

Several technologies have been developed in order to record the activity of the brain such as *ElectroEncephaloGraphy* (EEG) or *MagnetoEncephaloGraphy* (MEG), respectively recording the electrical and magnetic activity non-invasively, *ElectroCorticoGraphy* (ECoG), an invasive variant of the EEG where electrodes sit directly on the cortical surface, or *functional Magnetic Resonance Imaging* (fMRI), a neuroimaging technique that measures brain activity by detecting changes in blood flow. Other technologies such as *functional Near-Infrared Spectroscopy* (fNIRS), or *Microelectrode arrays* (MEAs) also exist. In the brain, the basic element generating a signal is a neuron, however it is not possible to detect the current generated by a single neuron using non-invasive technologies (invasive technologies do measure single-neuron activity). Thus, in order to be detected non-invasively, numerous neurons must be active at the same time. In this work, we exclusively restricted ourselves to EEG, let us thus dive into this technology.

EEG was first introduced by Hans Berger in 1924 during a surgical intervention (H. Berger

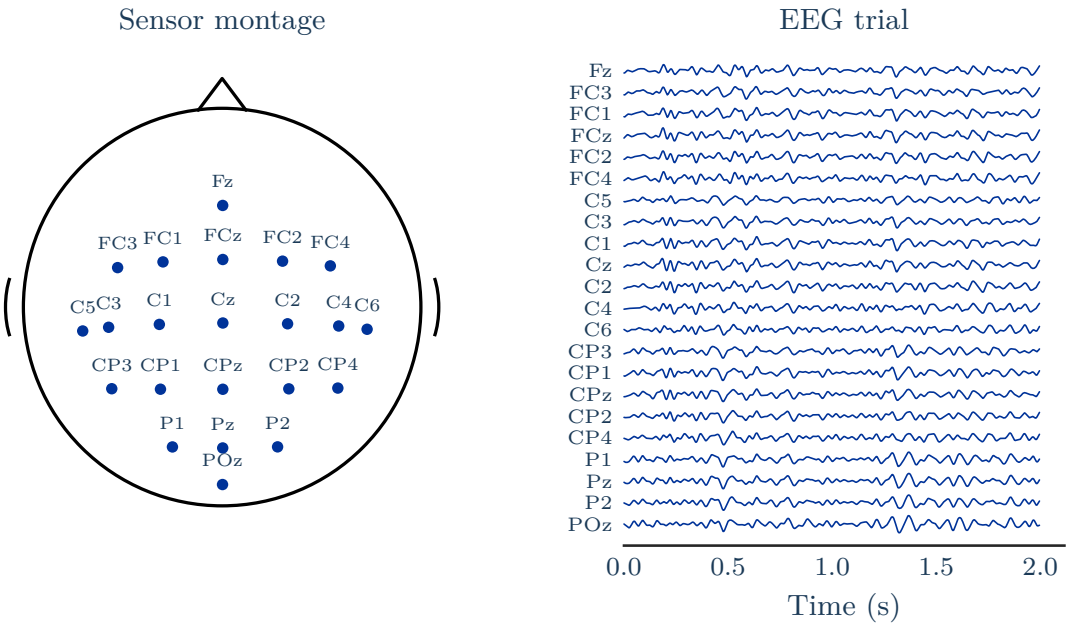


Figure 2.2. On the left is illustrated a standard montage for EEG based BCI. Each electrode has a predefined placement on the scalp. On the right, we show an example of a true recorded EEG. Each line corresponds to the electrical activity recorded by a given electrode. The montage corresponds to the one used for the dataset BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012) and the EEG is taken from this dataset.

1929). This technique consists in placing small electrodes immersed in conductive gel on the scalp of the patient, that each measure the electrical potential difference between a point on the scalp and a reference electrode. This difference in potential is created by the currents that propagate in the brain. The amplitude of the EEG signal varies between 1 to 100 μV (Teplan et al. 2002). By recording this voltage change over time, a time series is formed and is called an *ElectroEncephaloGram* (the acronym EEG is overloaded). Usually, several electrodes are placed at different positions on the scalp, ranging from three to several hundred (Oikonomou et al. 2016). The positions of the electrodes usually follow international standards, such as the 10-20 system. An illustration of the placement of the electrodes on the scalp as well as an example of an EEG is provided in Figure 2.2.

EEG has a high temporal resolution (recordings are usually sampled between 250 and 512 Hz, sometimes even higher, depending on the device used and its parameters) and its portability and low cost make it an appealing solution for a very wide range of applications. However, it has low spatial resolution, as the skull is less conductive than the brain and the scalp. Consequently, electrical currents attenuate and diffuse as they pass through the skin, the cerebrospinal fluid, and, most importantly, the skull. This results in spatial smoothing of the electrical potentials (Nunez and Srinivasan 2006). Moreover, EEGs are very sensitive to environmental noise such as electromagnetic interference or instrument artifacts (Symeonidou et al. 2018).

Intermission: BCI paradigms Before going further into the BCI pipeline given in Figure 2.1, we must describe the different approaches used in BCI, often called *paradigms*. A paradigm is the mix of what the subject is supposed to do during a BCI experiment as well as what is presented to the subject. We will describe the three main paradigms used in BCI: *P300*, *SSVEP*, and *Motor Imagery*. In the rest of this work, we will mainly focus on *Motor Imagery*.

- **P300** is a paradigm based on evoked potentials: the user is requested to focus on

something, like a specific letter on a screen, or a specific sound. When a letter is suddenly highlighted, thus creating a rare but task-relevant event, a specific weak EEG pattern (a positive deflection in the EEG) can be detected approximately 300ms after the event happens (the P300) (Farwell and Donchin 1988; Fazel-Rezai et al. 2012). This paradigm depends on the attention of the user, not only on an external physical stimulus.

- **SSVEP** stands for *Steady-State Visually Evoked Potentials* and is an electrophysiological response linked to a periodic visual stimulation. Typically, the subject fixates on a visual stimulus that is flashing at a given frequency, and the visual cortex will respond at the same frequency and its harmonics (Zhu et al. 2010; Norcia et al. 2015). Different stimulus frequency are associated to different mental commands. A variant of SSVEP is *c-VEP* for *code-modulated VEP* that uses pseudo-random sequences instead of pure frequencies, allowing for more simultaneous targets (Velut et al. 2026).
- **Motor Imagery** is the main paradigm studied in this thesis. In this paradigm, the subject imagines a movement of a limb (typically the left or right hand, the feet, the tongue) without executing it. This creates an event-related desynchronization (ERD) and an event-related synchronization (ERS) in specific frequency bands (the α and β bands, see Section 2.1.1) in the sensorimotor cortex (G. Pfurtscheller and C. Neuper 2001) and specific brain areas. This enables to discriminate the different imagined movements and to associate each of them to a different mental commands.

Preprocessing In order to extract the useful information from the recorded EEG, a proper preprocessing step is often mandatory. Indeed, the acquired EEG is often contaminated by noise or artifacts coming from non-neural signals such as eye blinks or muscle noise, from environmental noises such as the AC power line or the electromagnetic field noise coming from the surroundings, or even from the setup used such as the Analog-to-Digital conversion, the movements of cables or electrodes. Several studies have analyzed the effect of preprocessing on EEG data. For example, Bomatter et al. (2024) showed that basic artifact rejection improved model performance, while Delorme (2023) argue that many preprocessing steps might be useless or even harmful.

A typical preprocessing pipeline for EEG starts by filtering the frequency of interest. In the applications of interest in this thesis, we mainly focus on *motor imagery*, where the subject is asked to imagine moving a limb. For this application, the frequencies of interest range from 5 to 35 Hz (G. Pfurtscheller and Lopes da Silva 1999; G. Pfurtscheller and C. Neuper 2001), but it can be narrower, and we will see specific power bands in the *Feature Extraction* section below. Then, the acquired data is often re-referenced to a common reference, chosen for example to be the average of all the electrodes. This technique, called *Common Average Reference* (CAR), helps to minimize the influence of a specific reference electrode on the overall activity (Bertrand et al. 1985). It can happen that some electrodes malfunction for a part or for the entire recording, leading to a flat or excessively noisy signal. In this case, the channel should be removed during the preprocessing step. When conducting a cross-subject analysis, where the recordings of different subjects are analyzed together, it is however important to have the same setup across the subjects. Thus, several interpolation techniques have been developed in order to recover a malfunctioning channel such as the *ComImp* method from T. Nguyen et al. (2023) that expands the data to match the union of the set of channels, or physics-informed approaches presented in Mellot, Collas, S. Chevallier, Engemann, et al. (2024) based on spherical spline interpolation and field interpolation.

A crucial step is then artifact detection and removal, in order to identify and eliminate non-neural signals. The most commonly used algorithms for detecting artifacts are the *Independent Component Analysis* (ICA, Hyvärinen and Oja 2000) and the *Signal-Space Projection* (SSP, Uusitalo and Ilmoniemi 1997). ICA is a blind source separation technique that estimates

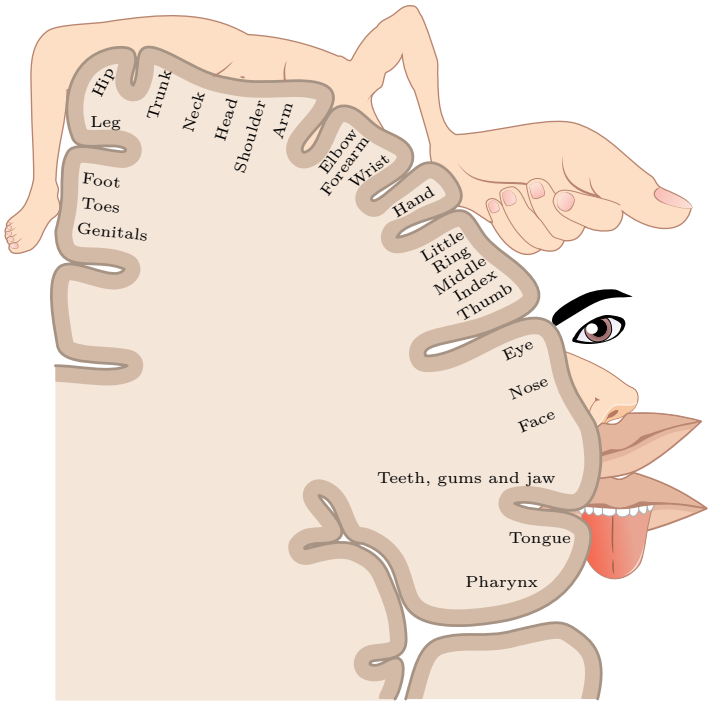


Figure 2.3. The different motor areas of the brain. The human body is laid across the brain in order to show the amount of primary somatosensory cortex dedicated to specific body regions. Adapted from "Sensory Homunculus" by OpenStax College, Popadius and Pereoptic, licensed under CC BY 3.0.

independent source signals from a set of recordings. In our case, the different sources are the activities of the potential artifacts (blinks, heartbeats, muscular activity in the jaw) mixed with the activity of the brain and the recordings are the different channels of the EEG. ICA will separate the different sources, and then re-construct the signal without the unwanted sources. SSP projects the data onto a lower-dimensional subspace orthogonal to the "noise directions". These directions are estimated from segments dominated by artifacts and from empty-room recordings acquired while no subject is present. The SSP is mainly used to remove environmental noises, while the ICA is usually used to remove biological noises. These techniques are mostly used offline, after the whole dataset has been recorded, and rarely online, when the recordings are made. We will see in Section 4.4 an outlier rejection tool that can be used online.

Finally, the signal is divided into small segments called epochs. The length of the epochs varies depending on the type of analysis that is conducted, and they are usually centered around the event of interest. For the motor imagery datasets that we will use in Chapter 4, the epochs are between 3 and 5 seconds.

Feature extraction One can model an EEG epoch as a matrix in $\mathbb{R}^{d\times t}$, where d is the number of electrodes and t the number of time samples. In a typical motor imagery dataset, such as the one proposed by Tangermann, K.-R. Müller, et al. (2012), the number of electrodes is $d = 22$, and the number of time samples in one epoch is $t = 1000$. Each epoch is thus a high-dimensional vector in $\mathbb{R}^{22\times 1000}$. The goal of feature extraction is to describe the EEG signal by a small set of variables that still capture the relevant information. Another advantage of having a small number of features is that they can be used *a posteriori* to interpret the decision made by the classifier.

As described in Chapter 7 of Clerc et al. (2016), there are three main sources of information that can be extracted from an EEG signal: spatial information, spectral (or frequency) information and temporal information. For the spatial information, one can use expert

knowledge on the brain, such as the motor areas of the brain which are illustrated in Figure 2.3. One can then select specific electrodes for the specific task one wants to solve. For example, when wanting to classify between imagined movements of the left hand versus imagined movements of the right hand, the central electrodes C3 and C4 (see Figure 2.2) are key.

One of the most common feature extraction procedures is to extract power in the relevant frequency bands. The classical frequency bands (Başar et al. 2001) are the θ band (4–7 Hz), the α band (8–13 Hz), the β band (14–30 Hz), and the γ band (above 30 Hz). For the paradigm of interest here, motor imagery, the α and β bands are the most relevant. Band power features represent the power (thus the energy), for a given frequency band of a given electrode. There exist several techniques to extract the band-power from an EEG (Brodu et al. 2011; Herman et al. 2008), and this feature is one of the gold-standard features used for BCI based on motor and mental imagery (Lotte, Bougrain, et al. 2018).

A very common method used to extract features from a band-pass filtered EEG is the *Common Spatial Pattern* algorithm (Ramoser et al. 2000; Blankertz, Tomioka, et al. 2008), whose goal is to learn spatial filters maximizing the discriminability between two classes. This is done by maximizing the variance of EEGs from one class while minimizing their variance from the other class. The variance of an EEG channel corresponds to its signal power, thus, this technique seeks discrimination based on power features (Ramoser et al. 2000). The optimal spatial filter w is obtained by maximizing the ratio of variance between the two classes:

$$w \in \arg\max_{w} \frac{w^\top \mathbf{E}_1 \mathbf{E}_1^\top w}{w^\top \mathbf{E}_2 \mathbf{E}_2^\top w},$$

where $\mathbf{E}_1$ and $\mathbf{E}_2$ are the EEG matrices of each class. The optimization problem is a *generalized Rayleigh quotient* and its solution is given by solving a generalized eigenvalue problem on the spatial covariance matrices of $\mathbf{E}_1$ and $\mathbf{E}_2$. The spatial filters used are the eigenvectors corresponding to the largest and the smallest eigenvalues in the above generalized eigendecomposition. Typically, six CSP filters are used, which correspond to the three largest and three smallest eigenvalues (Clerc et al. 2016). Once the filters have been computed, one can use the following feature:

$$\log(w^\top \mathbf{E}\mathbf{E}^\top w) \equiv \log(\mathrm{var}(w^\top \mathbf{E})),$$

that is, the log of the power of the signal spatially filtered with one of the CSP filters. This algorithm has provided high classification performance for BCI, while being numerically efficient and simple to implement. Another advantage of the CSP algorithm is that the learned spatial filters are interpretable. We will see in Chapter 7 that interpretability is important in BCI, and one can produce topographic maps from the filters learned by a CSP that are neurologically plausible (Haufe et al. 2014). For an example of such a heatmap, we refer to Figure 1 of Lotte and Guan (2011). Owing to these advantages, CSP was one of the most popular algorithms used to design BCI (Lotte and Guan 2011). However, CSP is known to overfit when few trials are available, or when the number of electrodes is high, since the covariance estimates become noisy. CSP is also sensitive to artifacts or outliers in the EEG. In Lotte and Guan (2011), the authors proposed different regularized versions of the CSP in order to mitigate these issues. In order to facilitate the selection of the frequency range (that is often done manually and subject-specifically), the authors of Ang et al. (2008) proposed *Filter Bank Common Spatial Pattern* (FBCSP) that computes several CSPs over several sub-bands and a feature-selection step picks the most discriminative ones across bands using a mutual information-based best individual feature (Battiti 1994). The CSP algorithm has also been linked to the Riemannian geometry of covariance matrices that will be of special focus in this work (see Section 2.1.4 and Barachant, Bonnet, et al. 2010a; Yger, Lotte, et al. 2015).

Finally, the temporal information of the EEG can be extracted by concatenating EEG samples from all channels. This is typically done after some downsampling and is useful when

wanting to detect or classify temporal variations in EEG signals' amplitude after a specific event or stimulus, such as P300-based BCI (Blankertz, Lemm, et al. 2011; Lotte 2014).

Lastly, the feature we will use the most in this thesis is not a vector feature, but rather a representation of EEGs based on a matrix, more specifically its *spatial covariance matrix*. We have already encountered the spatial covariance matrix in the CSP above. In the case of the CSP, the spatial covariance matrices are treated as linear objects, solving a linear generalized eigenproblem. We will see that, in order to respect the structure of covariance matrices, a non-flat Riemannian geometry should be used. This approach will be discussed in more depth in Section 2.1.4.

Classification After extracting the most relevant features of the EEGs, the next step is to classify them. Based on the review of Lotte, Bougrain, et al. (2018) and on the benchmark library for EEGs MOABB (Aristimunha, Carrara, et al. 2026), we group the classifiers used for EEG into three categories: classical ML algorithms, Riemannian classifiers and deep classifiers.

Up to the mid-2000s, as reviewed in Lotte, Congedo, et al. (2007), most classification algorithms used for BCI were not BCI specific. People mostly relied on popular linear classifiers such as *Linear Discriminant Analysis* (LDA, G. Pfurtscheller and Lopes da Silva 1999; Bostanov 2004; Garrett et al. 2003; Scherer et al. 2004), or *Support Vector Machine* (SVM, Rakotomamonjy et al. 2005; Garrett et al. 2003; Blankertz, Curio, et al. 2001). *Hidden Markov Models* were also explored for online classification (Obermaier, Guger, et al. 2001; Obermaier, Christa Neuper, et al. 2001) and multilayer perceptrons started to be used (Palaniappan 2005; C. W. Anderson and Sijercic 1996; Haselsteiner and Gert Pfurtscheller 2000). Later, adaptive versions of most of these algorithms were developed. By adaptive, we mean that the weights attributed to each feature are incrementally re-estimated and updated over time as new EEGs are available (Schlögl et al. 2010; Shenoy et al. 2006). More details on adaptive classifiers are available in Lotte, Bougrain, et al. (2018). These adaptive algorithms had the particularity of being explicitly tailored for BCI applications, rather than simply generic ML algorithms.

A major shift was initiated in 2010 by Barachant, Bonnet, et al. (2010b), when classifiers based on the covariance matrices of the EEG, together with the Riemannian geometry of covariance matrices were introduced. These algorithms will be at the heart of this manuscript, so we will not dive deeper here, however, we will motivate the use of covariance matrices in Section 2.1.4 and describe the Riemannian algorithms in Section 2.3.

The last leap forward in the BCI community is the use of deep architectures, tailored for EEG. Several deep architectures have been extended to deal with EEGs, such as Convolutional Neural Networks giving birth to *ShallowConvNet* and *DeepConvNet* (Schirrmeister et al. 2017) that, as Aristimunha Pinto (2026) says, "established the modern convolutional paradigm for EEG decoding". Several variants of this initial blueprint were then developed, such as *EEGNet* (Lawhern et al. 2018), or sleep-staging architectures (Perslev et al. 2021; Chambon et al. 2018). Because of its popularity and major success in natural language processing, attention-based models have also been developed for EEGs. A first example is *EEGConformer* (Yonghao Song et al. 2022), that combines CNNs, for local features, and a Transformer encoder, for global temporal context in a hybrid architecture. The *ATCNet* from Altaheri et al. (2022) combines a multi-head attention module (over the temporal dimension) with a temporal convolutional network block. These deep architectures are becoming more and more used, especially alongside an emerging direction of EEG foundation models, that require a tremendous amount of data to be trained (W.-B. Jiang et al. 2024; Kaifosh and Reardon 2025; Kaplan et al. 2020; El Ouahidi et al. 2025). A very complete and recent review of deep architectures can be found in Aristimunha Pinto (2026), together with the Python library *Braindecode* (Aristimunha, Guetschel, et al. 2026).

Method	Common features	Data needed	References
Classical ML			
LDA	Band power, CSP log-variance	Low	(G. Pfurtscheller and Lopes da Silva 1999; Garrett et al. 2003)
SVM (linear / RBF)	Band power, CSP log-variance	Medium	(Rakotomamonjy et al. 2005; Blankertz, Curio, et al. 2001)
Hidden Markov models	Temporal band-power sequences	Medium	(Obermaier, Guger, et al. 2001; Obermaier, Christa Neuper, et al. 2001)
Multilayer perceptron	Band-power features	Medium	(Palaniappan 2005; C. W. Anderson and Sijercic 1996)
Adaptive classifiers	Online-updated features	Low	(Schlögl et al. 2010; Shenoy et al. 2006)
Riemannian			
MDM	Covariance matrix	Low	(Barachant, Bonnet, et al. 2010b)
Tangent space + LDA	Covariance matrix	Low	(Barachant, Bonnet, et al. 2011)
Riemannian kernel SVM	Covariance matrix	Medium	(Barachant, Bonnet, et al. 2013)
Deep learning			
DeepConvNet	Raw epochs	High	(Schirrmeister et al. 2017)
ShallowConvNet / EEGNet	Raw epochs	Medium–High	(Schirrmeister et al. 2017; Lawhern et al. 2018)
Sleep-staging networks	Raw epochs	High	(Perslev et al. 2021; Chambon et al. 2018)
EEGConformer	Raw epochs	High	(Yonghao Song et al. 2022)
ATCNet	Raw epochs	High	(Altaheri et al. 2022)
SPDNet	Covariance matrix	High	(Huang and Van Gool 2017)
Foundation models	Raw epochs	Very high	(W.-B. Jiang et al. 2024; El Ouahidi et al. 2025)

Table 2.1. Three families of classifiers used for EEG decoding, following the taxonomy of Lotte, Bougrain, et al. (2018) and the MOABB benchmark (Aristimunha, Carrara, et al. 2026). Riemannian methods based on covariance matrices are described in Section 2.3.

An important caveat on the previous deep learning architectures used for BCI is that creating a proper benchmark comparing all the existing deep architectures is hard since, as Carrara (2024) notes "there is a noticeable lack of transparency regarding the selection of hyperparameters, as well as a limited availability of open-accessible code repositories". However, recent initiatives such as *MOABB* (Aristimunha, Carrara, et al. 2026) or *Braindecode* (Aristimunha, Guetschel, et al. 2026) try to bridge this gap and propose a reproducible benchmark for classifiers for EEG data.

A summary of the three families of classifiers used for EEG is given in Table 2.1.

2.1.2 Applications of BCIs

After discussing the main steps of the pipeline of a BCI, let us now focus on the applications of the BCIs. BCIs and more generally the recording of EEGs have a lot of different applications, and one of the main applications is the control of external devices through the paradigm of motor imagery. This paradigm can be used to control prosthesis (D. P. Murphy et al. 2017) or to control a wheelchair (Vanacker et al. 2007). However, BCIs have also allowed users to write on a virtual keyboard using only their brain, with a speed of around 7 to 8 characters per minute (Blankertz, Dornhege, et al. 2010), or to control a cursor moving on a screen (Wolpaw et al. 1991). Recently, a video game player, Perri Karyal, built a BCI in

order to play the video game *Elden Ring*[1], by assigning different mental tasks to different actions in the game, using a 16-electrode EEG cap. Similarly, BCIs have been used together with augmented reality (Si-Mohammed et al. 2018), allowing the user to navigate menus without relying on any movement. Motor-imagery-based BCIs have also been used in sports, for example for mental imagery training of athletes, and can be coupled with virtual reality for a more immersive experience (Jeunet, Hauw, et al. 2020; M. Pereira et al. 2018).

More generally, EEG can be used to detect the concentration level and mental workload of a subject (Matthews et al. 2015; Sharma and Ahirwal 2024; Cimarosto et al. 2026). This was also used by Trocellier, Meunier, et al. (2025) to control a small kart's speed using their level of focus and mental engagement. Players were instructed to perform mental tasks, such as solving mathematical equations or translating a text in order to increase the speed of the kart. In Haseeb et al. (2025), the authors used EEG to detect pilot mental workload in real flight scenarios.

In a clinical setting, EEG can be used to diagnose and monitor epilepsy (Chowdhury et al. 2015; M.-P. Hosseini et al. 2016). EEG can be recorded over several hours of sleep in order to detect the different stages of sleep and can thus help to detect sleep disorders such as sleep apnea, narcolepsy, and insomnia (Gnassounou et al. 2026; R. Behzad and A. Behzad 2021). As we will see in Chapter 6, EEG can be used during a surgery to assess the level of anesthesia of the patient and avoid accidental awareness during general anesthesia (J.J. Pandit et al. 2014; Almeida 2015). Another application is the rehabilitation process of some cases of stroke (Silvoni et al. 2011; Ramos-Murguialday et al. 2013). In this case, BCIs are used to detect movement intentions in stroke patients who have a paralyzed limb, in order to guide them to use the damaged brain area and stimulate brain plasticity in that area, and therefore, recovery. BCIs have also been used to help the rehabilitation of patients suffering from an impairment to understand or produce language, for example after a stroke (Musso et al. 2022). Other applications of EEG based BCI in the medical field are given in the review of X.-Y. Liu et al. (2025).

2.1.3 Challenges of BCIs

BCI is a very promising technology that could enable controlling devices directly using our brains, without relying on any intermediate tools currently used. However, BCIs face several challenges in order to work properly, and the goal of this section is to describe some of them. For now, the vast majority of promising BCI applications are evaluated only within a single recording session, in a single context (typically a well-controlled laboratory environment), and for a single application. This contrasts sharply with the ultimate goal of BCI research: developing reliable systems that users can seamlessly integrate into their daily life. Bridging this gap remains a major challenge.

One of the first limitations of BCIs is that, most of the time, these systems remain confined to controlled laboratory environments and are rarely used in the real world, outside the laboratory. A first limitation comes from the technology itself: even if EEG can be considered "light-weight" compared to MEG or fMRI, it requires special devices that are not always usable in everyday life. For example, typical EEG caps require using conductive gel between the electrodes and the scalp, in order to decrease the contact impedances. An alternative to using wet electrodes is to rely on dry electrodes, with the risk of degrading the quality of the recorded signal (Chi et al. 2010; Mullen et al. 2015). In addition to the technological problem posed by the EEG recording device, during a typical BCI experiment, the subject is sitting on a chair, focused on the task they want to achieve (Lotte, Dreyer, et al. 2025). In contrast, in a real-world scenario, the subject will move, generating artifacts that need to be preprocessed and, most likely, removed (Gramann et al. 2010). However, in Debener

[1] See https://www.youtube.com/watch?v=rIbfNUA5pWk, accessed July 3rd, 2026.

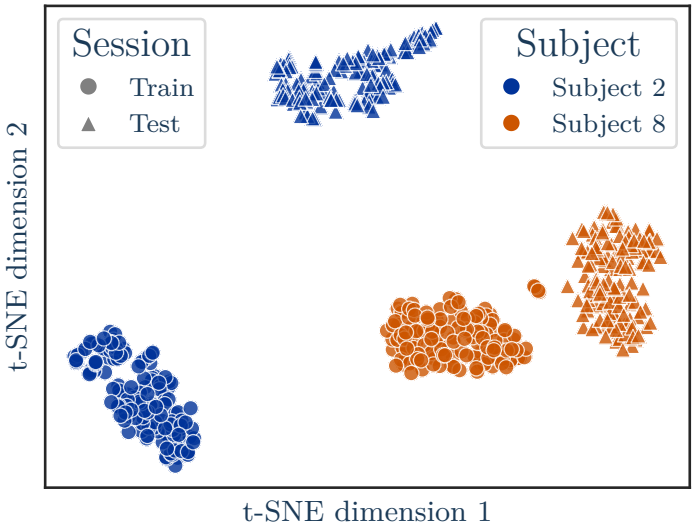


Figure 2.4. An illustration of the EEG variabilities of the dataset BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). We computed the t-SNE embedding for the covariance matrices estimated from the EEGs of subject 2 and 8 of BNCI2014001. We can clearly see the inter-subject variabilities (the two colors, each representing one subject, are very well separated), as well as the intra-subject variabilities (for each subject, two sessions were recorded and the marker type corresponds to the session, the two sessions for each subject are also well separated).

et al. (2012), the authors showed that it was possible to achieve good performances (at least comparable to a controlled setting) with a low-cost, small, and wireless EEG system, while the subject was slowly walking on Oldenburg campus. This study challenges the assumption that EEG only works in the lab.

Apart from technological difficulties and artifacts arising from real movements, there is another major challenge facing BCIs that will be at the heart of this thesis: EEGs are **highly** non-stationary: the signal distribution shifts continuously due to electrode drift, fatigue, mental state changes, and context variation (Stam 2005; Popivanov and Mineva 1999). These variabilities can be classified into two categories: *intra-subject variabilities*, variabilities that occur within a given subject between sessions or contexts and *inter-subject variabilities*, variabilities that appear between subjects (Fairclough and Lotte 2020). These variabilities are illustrated in Figure 2.4, where we have plotted a two-dimensional t-SNE embedding[2] (Maaten and Hinton 2008) for subject 2 and subject 8 of the dataset BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). This plot gives us a clear illustration of the two types of variabilities introduced above: each color corresponds to a subject, and we can see that the two subjects are very clearly separated, illustrating the inter-subject variabilities. Moreover, this dataset has been recorded during two sessions, and each marker corresponds to a session. We can also see that, within a given subject, the two sessions are separated, illustrating the intra-subject variabilities.

The intra-subject variabilities are the variabilities that occur within a given subject. These variabilities can be subdivided into two sub categories: *within-session variabilities*, which are variabilities that happen during a given session and *cross-session variabilities*, appearing from one session to another. A session is a continuous period of data acquisition performed without substantial interruption, under a fixed experimental protocol, usually on a single day. During a given BCI session, the mental state of the subject can change due to fatigue, adaptation or fluctuations in attention. In Grosse-Wentrup and Schölkopf (2013), the authors display the performance of a classifier during a 20-minute left versus right hand motor imagery experiment. They start by showing that, for a specific subject, during the first nine minutes,

[2]We will propose in Chapter 7 a fully Riemannian version of the t-SNE algorithm.

the classifier always predicts the correct label. However, during the following seven minutes, a large proportion of trials are not correctly classified, before a slight positive drift is observed at the end of the session. This shows that for a given command (here a motor imagery command), the recorded EEG is not the same at t and at $t + \Delta t$. This is even more visible when going from one session to another. Indeed, the same task performed by the same user on different days or at different times of the day will yield different feature distributions (Grosse-Wentrup and Schölkopf 2013; Saha and Baumert 2020; Kasahara et al. 2015). The known sources of variabilities are the users' mental state, due to fatigue or stress (Roy, Bonnet, et al. 2013; Talukdar et al. 2020), the context (Benaroch et al. 2021) or environmental noise (Dehais et al. 2019) to list a few. For example, in a cross-day workload EEG classification competition organized by Roy, Hinss, et al. (2022), many competitors obtained good within-day accuracies ($> 80\%$), but very poor cross-day accuracies (< 50 %) on a 3-class problem.

If, within a given subject, variabilities make it hard to build reliable BCIs, one can easily imagine that building a BCI that works on several subjects is an even harder task. This is due to the inter-subject variabilities. Each subject has their own cognitive style and could choose among several strategies to perform the same task (Saha and Baumert 2020; Seghier and Price 2018). In Jeunet, N'Kaoua, et al. (2016), the authors give three categories for motor imagery based BCI performance predictors: the users' relationship with technology, their attention and their spatial abilities. The emotional and cognitive state of the user has been shown to influence the BCI performance in terms of control accuracy (Nijboer et al. 2008), as well as attention (Grosse-Wentrup and Schölkopf 2012) and motivation (Hammer et al. 2012). The brain structure itself can influence the performance of BCI: in Halder et al. (2013), the authors analyze structural data from anatomical scans and DTI (Diffusion Tensor Imaging) and identify discriminating structural brain features that correlate with individual BCI performances. In Kasahara et al. (2015), the authors showed that neuroanatomical features such as gray matter volume are associated with BCI performance. These inter-subject variabilities are very visible in large-scale studies over several subjects such as the ones conducted in MOABB (Aristimunha, Carrara, et al. 2026) or in Sannelli et al. (2019). The latter study shows that even within a controlled lab setting, some subjects never achieve reliable control regardless of training duration. This is called *BCI deficiency* or *BCI illiteracy* and is another challenge when designing BCIs. This notion of BCI illiteracy is criticized, as in (Thompson 2019), where the author describes it as "an inadequate concept for explaining difficulty that users face in operating BCI systems".

The last challenge of BCI that will be discussed here is the cost of collecting data. Setting up a BCI experiment can be very costly and requires specific authorizations (Lotte, Dreyer, et al. 2025), thus acquiring data from a lot of subjects in a lot of different contexts can be cumbersome. Data scarcity is an important challenge for the BCI community, especially with the development of deep learning algorithms that require a lot of data to be correctly trained (see Table 2.1). Some very large datasets are starting to appear, such as Gifford et al. (2022) or J. Xu et al. (2025), that are made for modeling human visual object recognition, with respectively 82,160 trials and 83,520 trials per subject. Foundation models especially require a huge amount of data to be trained, and in El Ouahidi et al. (2025), the authors aggregated 92 different datasets to build a dataset consisting of more than 24,000 subjects. The model uses a masked autoencoding objective with a novel 4D positional encoding that handles arbitrary electrode configurations.

At a smaller scale, when the features used are the covariance matrices, the sample covariance estimation is unreliable when the number of samples is less than the number of electrodes used. In practice, this is solved by regularizing the estimated covariance matrices (Ledoit and Wolf 2004; Y. Chen et al. 2010). More on this in Section 2.1.4. Other solutions to data scarcity are data augmentation techniques, relying on the geometry of covariance matrices such as Kalunga et al. (2015) or using deep learning techniques such as GAN (Hartmann et al.

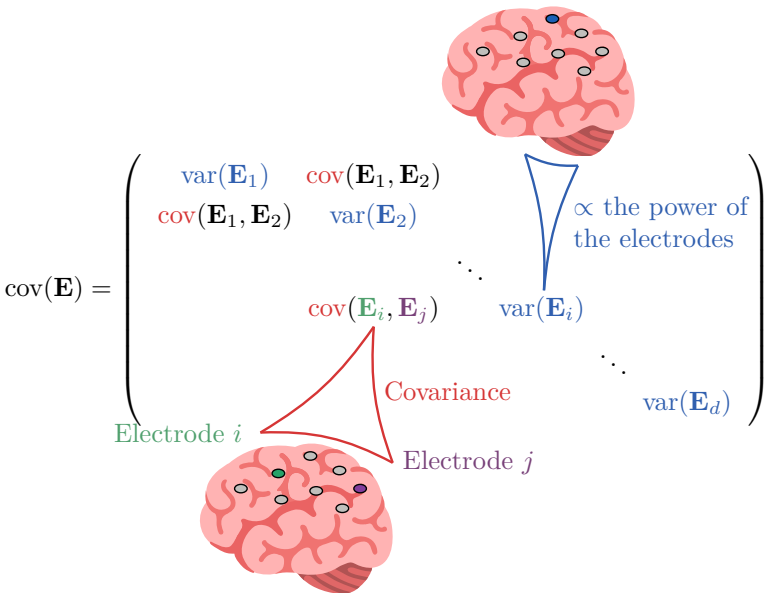


Figure 2.5. Illustration of a covariance matrix estimated from an EEG. The diagonal elements are proportional to the power of each electrode, and the non-diagonal elements give the covariance between two given electrodes.

2018) or diffusion models (Aristimunha, Camargo, et al. 2023). We refer to Tangermann, S. Chevallier, et al. (2025) for various techniques to train classification or regression machine learning models with small BCI datasets.

The question of handling intra-subject variabilities in BCI was at the origin of this PhD as it was funded by the French National Research Agency for project PROTEUS[3], whose goal is to measure, understand and model computationally the variabilities affecting BCIs. Then, based on these models, the aim is to design EEG machine learning classifiers specifically able to handle such variabilities.

2.1.4 Using covariance matrices and their Riemannian geometry to build better BCIs

We have described in Section 2.1.1 the different types of features used in BCI. The one we will be using throughout this work is the *covariance matrix*, estimated from the EEG. These covariance matrices are very structured mathematically, and need to be handled with special attention. We now describe why this feature is important and start to motivate the use of a specific geometry.

2.1.4.1 Motivation

In Section 2.1.1, we saw that one of the most commonly used features from EEG are the powers in specific frequency bands. The simplest case is when considering the vector of per-channel band powers. Let us consider that we have access to a continuous[4] time dependent EEG denoted $\mathbf{E}\colon t \in \mathbb{R}^+ \mapsto \mathbb{R}^d$ recorded using d electrodes. This EEG is considered as a random process and let us assume that it is centered, stationary (meaning that its statistics do not change over time) and ergodic (see Chapter 6 of Gray 2009). A first important remark is that, when discussing the variabilities that challenge BCI in Section 2.1.3, we said that the EEG signal was highly non-stationary. This is true when we consider EEG recorded during

[3] Grant ANR-22-CE33-0015- 01.
[4] In practice, we simply have a discrete version of the continuous signal, sampled at a certain frequency.

a long period of time, several minutes or several sessions. Here, our goal is to estimate the covariance matrix of one EEG epoch. An epoch is a fixed-length segment of EEG time-locked to a reference event such as a visual cue or a stimulus onset. The epochs we consider here last around several seconds (from 2 to 5 seconds in the datasets considered in the thesis). In the following, we will assume that, during an epoch, the EEG is stationary. Under these assumptions (centered, stationary and ergodic), the power of the i-th channel is:

$$\mathbb{E}[|\mathbf{E}_i(t)|^2],$$

which is exactly the variance of the single channel EEG $\mathbf{E}_i$. Building on this, the next features one could be interested in are the covariances between the different channels. The covariance $\text{cov}(\mathbf{E}_i, \mathbf{E}_j)$ between channel i and channel j gives us a measure of the joint variability of the two channels. All of this information can be summarized in the *covariance matrix*:

$$\text{cov}(\mathbf{E}) = \begin{pmatrix} \text{var}(\mathbf{E}_1) & \text{cov}(\mathbf{E}_1, \mathbf{E}_2) & \cdots & \text{cov}(\mathbf{E}_1, \mathbf{E}_d) \\ \text{cov}(\mathbf{E}_2, \mathbf{E}_1) & \text{var}(\mathbf{E}_2) & \cdots & \text{cov}(\mathbf{E}_2, \mathbf{E}_d) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(\mathbf{E}_d, \mathbf{E}_1) & \text{cov}(\mathbf{E}_d, \mathbf{E}_2) & \cdots & \text{var}(\mathbf{E}_d) \end{pmatrix}.$$

Since the covariance between two one-dimensional signals is symmetric ($\text{cov}(\mathbf{E}_i, \mathbf{E}_j) = \text{cov}(\mathbf{E}_j, \mathbf{E}_i)$), the covariance matrix $\text{cov}(\mathbf{E})$ is a symmetric matrix containing only $d(d+1)/2$ different coefficients. Moreover, computing the trace of the covariance matrix $\text{cov}(\mathbf{E})$ will give us the total power of the signal. The contributions to the different elements of the covariance matrix are illustrated in Figure 2.5. This between-electrode covariance is also referred to as *spatial covariance*.

One could ask how the variance and covariance can be related to physiological features other than the power. A core paper on BCI is G. Pfurtscheller and Lopes da Silva (1999), in which the authors describe how hand motor imagery produces a desynchronization in the α (8-13 Hz) and β (14-30 Hz) frequency bands at the electrodes C3 for right hand and C4 for left hand (see Figure 2.2, they are placed at the top on the scalp). This desynchronization corresponds to a decrease in power in a given frequency band resulting in a decrease in the variances of electrodes C3 and C4. This is why motor imagery can be identified from the (co)variance structure of these electrodes, motivating the use of spatial covariance matrices as features. This interpretability question for covariance matrices will be addressed more thoroughly in Chapter 7.

2.1.4.2 Estimating covariance matrices

Now that we are convinced that using the covariance matrix of an EEG as a feature can be useful, we will see how this covariance matrix can be computed in practice. In the above section, we have assumed the EEG to be stationary during an epoch. Moreover, as it is done in Engemann and Gramfort (2015), we will assume that the EEG follows a centered Gaussian distribution. This assumption can be supported by the fact that each EEG channel is the superposition of contributions from many thousands of neurons. Thus, by the Central Limit Theorem (Theorem 5.12 of Wasserman 2004), the sum tends toward Gaussianity regardless of the individual neuron firing statistics. Moreover, classical preprocessing pipelines as described in Section 2.1.1 include a band-pass filter, so the filtered EEG has a mean that is close to zero. Another justification of the Gaussianity of EEG given by Engemann and Gramfort (2015) is that, thanks to the linearity of Maxwell's equations describing electrical and magnetic fields, EEG are obtained through a linear mixing of brain sources and additive noise. Assuming the source amplitudes and the noise are Gaussian, the recorded EEG is also Gaussian.

Using the Gaussian assumption of EEGs, one can first estimate the covariance matrix using the *sample covariance matrix*. Given an EEG $\mathbf{E} \in \mathbb{R}^{d \times t}$ recorded using d electrodes

during t time samples, the sample covariance matrix $\hat{\Sigma}_{\text{SCM}}$ is

$$\hat{\Sigma}_{\text{SCM}} = \frac{1}{t-1}\mathbf{E}\mathbf{E}^{\top}. \tag{2.1}$$

This estimator has some nice theoretical properties such as its unbiasedness and its consistency, however when the number of time samples t is small compared to the number of electrodes d, this estimation can be unstable and numerically ill-conditioned. Thus, several regularized estimators have been proposed to solve this issue. In BCI, the most used ones are *shrinkage estimators* that regularize the sample covariance matrix by pulling it toward a structured target matrix $\boldsymbol{T}$:

$$\hat{\Sigma}_{\alpha} = (1-\alpha)\hat{\Sigma}_{\text{SCM}} + \alpha\boldsymbol{T},$$

where $\alpha \in [0,1]$ is a shrinkage coefficient. The most common target is the scaled identity $\boldsymbol{T} = \frac{\text{tr}(\hat{\Sigma}_{\text{SCM}})}{d} I_d$, which pulls eigenvalues toward their mean. This regularization term is interpretable: it adds a fraction of the average per-channel power to each diagonal entry, while preserving the overall power scale. Several ways of choosing the shrinkage coefficient α have been proposed. In Ledoit and Wolf (2004), the authors proposed an analytical, data driven formula for α that minimizes the expected Frobenius loss between the regularized covariance matrix and the true one. When the data is assumed to be Gaussian, the *Oracle Approximating Shrinkage* (OAS) proposed in Y. Chen et al. (2010) tends to outperform Ledoit-Wolf.

Other covariance estimators have been developed when the data is not Gaussian. These robust estimators are built to deal with heavy-tailed data or data containing outliers. The most famous ones are the fixed point estimator *Tyler's M-Estimator* proposed in Tyler (1987) and the *Huber's M-Estimator* (Huber 1981) that interpolates between the sample covariance and Tyler's estimator. A last covariance estimator is the *Minimum Covariance Determinant* from Rousseeuw (1984). This estimator tries to find a proportion of "good" observations, computes their empirical covariance matrix before rescaling it to compensate for the selection of observations done beforehand.

2.1.4.3 Learning with covariance matrices

We have seen why covariance matrices are useful and how to estimate them from a given EEG trial. Going back to the BCI pipeline illustrated in Figure 2.1, we now need to tackle the classification part when the features used are the covariance matrices. A first possibility would be to simply consider the feature vector of the variances and the covariances of the EEG trial, and use classical learning algorithms such as the ones described in Section 2.1.1. However, the covariance matrix does not only store the variance and covariance information, it also has a highly structured mathematical nature. Indeed, a covariance matrix is *Symmetric Positive Definite* (SPD). The set of SPD matrices is a subset of the set of symmetric matrices and considering simply the vector of variances and covariances amounts to treating the covariance matrix simply as a symmetric matrix, thus losing a significant amount of information. From a mathematical point of view, we will quantitatively describe in Section 2.2.2 why forgetting the *positive definiteness* of a covariance matrix, simply treating it as a symmetric matrix and applying the Euclidean geometry is not a good idea and can deteriorate the performances of a learning algorithm.

Since the 2010's, a new way of manipulating these covariance matrices (and more generally SPD matrices) has emerged in the BCI literature. This promising approach is based on a well-known concept for mathematicians: using the Riemannian geometry of the set of SPD matrices (Pennec 2020). Alexandre Barachant was the first one to use this curved geometry for dealing with covariance matrices estimated from EEGs (Barachant, Bonnet, et al. 2010b). This Riemannian geometry gives us a new way of measuring distances between covariance matrices, and, as a lot of learning algorithms are based on geometric criteria, changing the

intrinsic geometry also affects the learning algorithms. This curved geometry will be described rigorously in Section 2.2. A lot of algorithms based on this Riemannian geometry have been developed over the past 15 years (Congedo et al. 2017; Yger, Berar, et al. 2017) and Riemannian methods are now state of the art in many BCI tasks. These methods have remarkably won several BCI competitions (Roy, Hinss, et al. 2022) and figure among the best pipelines in the MOABB leaderboard (Aristimunha, Carrara, et al. 2026).

Our goal here is to motivate the use of this geometry for the specific application of BCI. Several specific geometries have been designed for the set of SPD matrices, and some of them will be presented in Section 2.2. The main one we will focus on in this thesis is based on the *Affine Invariant Riemannian Metric* (AIRM, Pennec 2020). The specificity of this geometry is that geometric operations such as the distance are *invariant by affine transformations*. More precisely, let us consider Σ_1 and Σ_2 two covariance matrices. We denote by δ the AIRM distance defined in Definition 2.13. This distance being affine invariant means that

$$\delta(\boldsymbol{A}\Sigma_1\boldsymbol{A}^\top, \boldsymbol{A}\Sigma_2\boldsymbol{A}^\top) = \delta(\Sigma_1, \Sigma_2),$$

for any invertible matrix $\boldsymbol{A}$. This property is interesting for BCI applications because of the *EEG forward model* described in Lecaignard and Mattout (2015): the relationship between the real source signals $\mathbf{S}(t) \in \mathbb{R}^p$ and the potential $\mathbf{E}(t) \in \mathbb{R}^d$ recorded by the electrodes on the scalp is governed by the following equation (see Equation 3.2 of Lecaignard and Mattout 2015):

$$\mathbf{E}(t) = \boldsymbol{A}\mathbf{S}(t) + \varepsilon(t),$$

where $\boldsymbol{A}$ is called the *lead-field matrix* and encodes the anatomical and biophysical information of a subject (e.g. head geometry, skull conductivity, electrode placement) and ε models an additive measurement noise. Then, estimating the covariance matrix using the sample covariance matrix given by Equation (2.1), one has that

$$\Sigma_{\mathbf{E}} = \boldsymbol{A}\Sigma_{\mathbf{S}}\boldsymbol{A}^\top + \Sigma_\varepsilon,$$

where $\Sigma_{\mathbf{S}}$ and Σ_ε are respectively the covariance matrices of $\mathbf{S}$ and ε. Thus, any change in the mixing matrix $\boldsymbol{A}$, such as electrode repositioning, changes in head geometry, or inter-subject anatomical variability, transforms $\Sigma_{\mathbf{E}}$ by a congruent action $\Sigma_{\mathbf{E}} \mapsto \boldsymbol{B}\Sigma_{\mathbf{E}}\boldsymbol{B}^\top$ for some matrix $\boldsymbol{B}$. Assuming that this transformation is invertible ($\boldsymbol{B}$ is invertible), the affine invariance of the AIRM metric implies that the geometry around $\Sigma_{\mathbf{E}}$ is the same as the geometry around $\boldsymbol{B}\Sigma_{\mathbf{E}}\boldsymbol{B}^\top$. The transformation is not visible to the AIRM geometry. No Euclidean metric on the space of symmetric matrices possesses this property. Thus, as Congedo et al. (2017) frames it, there is an "equivalence between electrode space and source space". They also show that the Riemannian mean computed using the curved geometry is more robust to outliers than the Euclidean mean.

A final argument in favor of using the Riemannian geometry of covariance matrices is related to one of the challenges given in Section 2.1.3: the scarcity of the data. Indeed, we saw that EEG data was expensive to collect, thus one should make the most out of the available data. As shown in Table 2.1, Riemannian classifiers do not require a huge amount of data in order to be trained. Thus, it is a good middle ground between classical Euclidean methods (performing badly) and deep learning methods (requiring a lot of data).

2.2 Mathematical background

As described in Section 2.1.4, the covariance matrix is an interesting and useful feature to use when building BCIs. As we saw, these mathematical objects need to be treated with special attention. In this section, we give the mathematical tools that will be needed to apprehend the rest of this manuscript. After recalling some important properties of *Symmetric*

Positive Definite (SPD) matrices, we motivate the use of Riemannian geometry for the set of SPD matrices. Then, we will describe different geometries that have been used for SPD matrices, with a main focus on the *Affine Invariant Riemannian geometry* (Pennec 2020). The advantages and disadvantages of each geometry will be discussed, with some small numerical experiments to compare their numerical complexities.

We omit the proofs of the following results, but provide appropriate references to the relevant literature for each case. In the following, d is an integer greater than one, as in practice, we will not study 1×1 SPD matrices[5]. We will focus our study on **real** matrices, thus all matrices in what follows will have real coefficients.

2.2.1 Matrix properties

Let us start by defining the core object of this thesis, that is a *Symmetric Positive Definite* matrix. We then give some of its basic characterizations and several classical properties. For proofs, we refer to the book of Bhatia (2007), or chapter 7 of Horn and Johnson (2012). For French readers, the author warmly recommends Caldero and Germoni (2016), specifically chapter 6.

Definition 2.1. A matrix $\boldsymbol{X}$ of size $d \times d$ is said to be *Symmetric Positive Definite* (SPD) if it is symmetric ($\boldsymbol{X} = \boldsymbol{X}^\top$) and positive definite i.e.

$$\forall\, v \in \mathbb{R}^d \setminus \{0\},\ v^\top \boldsymbol{X} v > 0.$$

We denote by $\mathcal{P}_d$ the set of all $d \times d$ SPD matrices. Let us recall that the set $\mathcal{S}_d$ of $d \times d$ symmetric real matrices is a vector space of dimension $d(d+1)/2$. Moreover, a crucial result is the spectral theorem that gives an orthonormal diagonalization of any symmetric matrix $\boldsymbol{X}$.

Theorem 2.2 (Spectral theorem)**.** *Let $\boldsymbol{X}$ be a $d \times d$ symmetric matrix. Then, there exists an orthogonal matrix $\boldsymbol{U}$ ($\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\top = I_d$) and a diagonal matrix $\boldsymbol{\Lambda}$ such that*

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top.$$

Moreover, the coefficients of $\boldsymbol{\Lambda}$, thus the eigenvalues of $\boldsymbol{X}$, are real numbers.

The previous theorem will be heavily used in the following without explicit reference. A first example of its application is the following characterization of SPD matrices where the spectral theorem is a key part of the proof.

Proposition 2.3. *Let $\boldsymbol{X}$ be a $d \times d$ symmetric matrix. $\boldsymbol{X}$ is positive definite if, and only if, all the eigenvalues of $\boldsymbol{X}$ are strictly positive.*

The previous proposition helps us define the unique square root of an SPD matrix.

Theorem 2.4. *Let $\boldsymbol{X} \in \mathcal{P}_d$. There exists a unique square root $\boldsymbol{R}$ of $\boldsymbol{X}$, that is a matrix $\boldsymbol{R} \in \mathcal{P}_d$, such that $\boldsymbol{R}^2 = \boldsymbol{X}$. It will be denoted $\boldsymbol{X}^{1/2}$.*

One can now focus on the structure of the set $\mathcal{P}_d$ of SPD matrices and how it is linked to other important sets, such as the set $\mathcal{S}_d$ of symmetric matrices, the set $\mathrm{GL}(d)$ of invertible $d \times d$ matrices or the set $\mathrm{O}(d)$ of $d \times d$ orthogonal matrices. A first important link is the following inclusion that will be used when defining a manifold structure on $\mathcal{P}_d$.

Proposition 2.5. *$\mathcal{P}_d$ is an open convex cone of $\mathcal{S}_d$.*

The link between $\mathcal{P}_d$ and $\mathcal{S}_d$ is actually stronger: both sets are diffeomorphic thanks to the following theorem.

[5] However, the geometry described still stands for this case.

Theorem 2.6. *The matrix exponential mapping* $\exp\colon \mathcal{S}_d \to \mathcal{P}_d$ *is a diffeomorphism.*

As the matrix exponential mapping $\exp\colon \mathcal{S}_d \to \mathcal{P}_d$ is a diffeomorphism, its inverse function is correctly defined and is continuous. This inverse mapping is called the matrix logarithm and is denoted $\log\colon \mathcal{P}_d \to \mathcal{S}_d$.

Remark 2.7. **Caution!** In the following manuscript, the matrix exponential and logarithmic mapping will always be written with lowercase letters: $\exp, \log$. They should not be confused with their Riemannian counterparts that will start with a capital letter: $\mathrm{Exp}, \mathrm{Log}$.

Let us now note a key stability property of SPD matrices under congruence transformations.

Proposition 2.8 (Congruence invariance)**.** *Let* $\boldsymbol{X} \in \mathcal{P}_d$ *and* $\boldsymbol{A} \in \mathrm{GL}(d)$*. Then* $\boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^\top \in \mathcal{P}_d$*.*

This proposition shows that the map $(\boldsymbol{A}, \boldsymbol{X}) \mapsto \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^\top$ defines a (transitive) group action of $\mathrm{GL}(d)$ on $\mathcal{P}_d$. The stabilizer of the identity matrix I_d under this action (i.e. the set of matrices $\boldsymbol{A}$ such that $\boldsymbol{A}\boldsymbol{A}^\top = I_d$) is precisely $\mathrm{O}(d)$, which gives the following proposition.

Proposition 2.9.

$$\mathcal{P}_d = \mathrm{GL}(d)/\mathrm{O}(d).$$

2.2.2 What geometry for SPD matrices?

Now that we have given some properties of SPD matrices, we want to see what geometry is best suited for studying the set $\mathcal{P}_d$ of SPD matrices. As suggested by the definition of an SPD matrix, the set $\mathcal{P}_d$ of SPD matrices is included in the space $\mathcal{S}_d$ of symmetric matrices. This set $\mathcal{S}_d$ is a Euclidean space when equipped with the Frobenius inner product $\langle \cdot, \cdot \rangle_F$, defined for $\boldsymbol{X}$ and $\boldsymbol{Y}$ two symmetric matrices by

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle_F = \mathrm{tr}(\boldsymbol{X}^\top \boldsymbol{Y}) = \mathrm{tr}(\boldsymbol{X}\boldsymbol{Y}).$$

Thus, one could consider using this simple flat geometry when dealing with SPD matrices. A classical geometric operation consists in interpolating between two points, here between two SPD matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$. As stated by Proposition 2.5, $\mathcal{P}_d$ is convex, thus the Euclidean interpolation between $\boldsymbol{X}$ and $\boldsymbol{Y}$, the straight line $t \in [0,1] \mapsto (1-t)\boldsymbol{X} + t\boldsymbol{Y}$, will remain well within the set $\mathcal{P}_d$. However, if one tries to extrapolate this straight line (i.e., take $t < 0$ or $t > 1$), at one point, the extrapolated point will reach the boundary of $\mathcal{P}_d$ and leave the cone of SPD matrices. Let us illustrate this with an example.

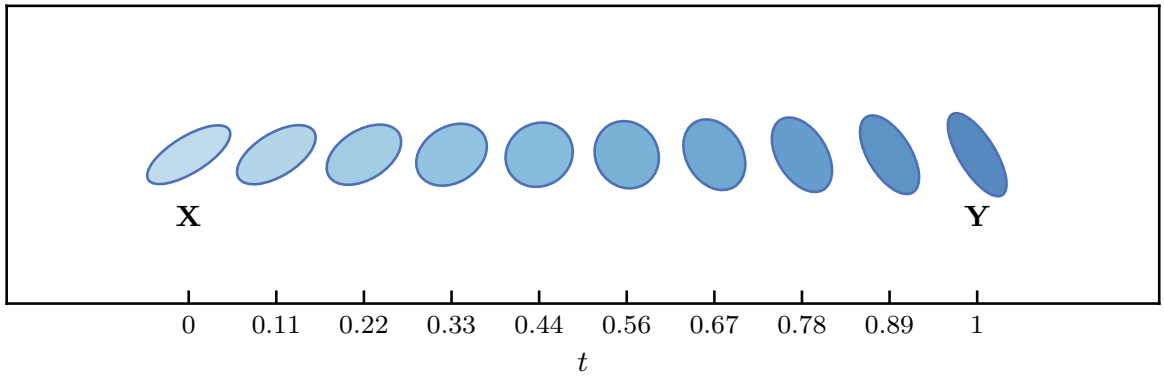
Example 2.10. Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be the two following matrices

$$\boldsymbol{X} = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix} \text{ and } \boldsymbol{Y} = \begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix}.$$

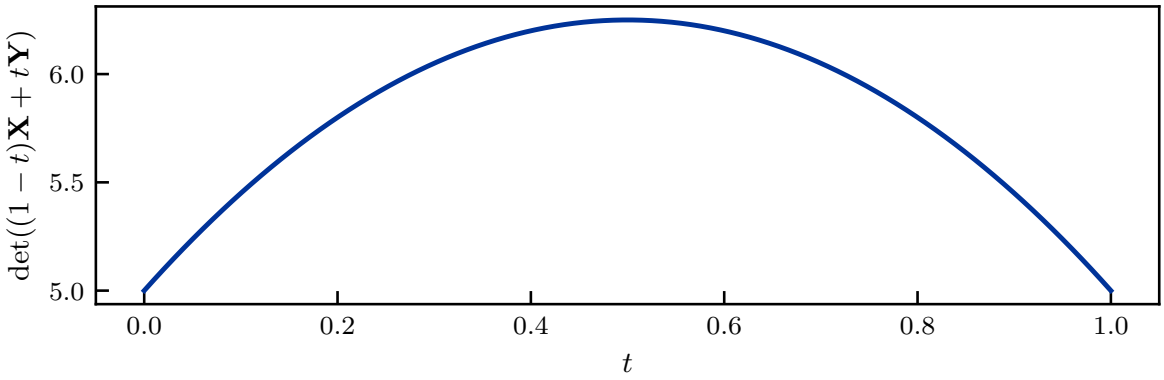
The matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are symmetric, and both their eigenvalues are $\frac{1}{2}(5+\sqrt{5})$ and $\frac{1}{2}(5-\sqrt{5})$ which are strictly positive. Thus, thanks to Proposition 2.3, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are SPD matrices. Let us consider the Euclidean interpolation between $\boldsymbol{X}$ and $\boldsymbol{Y}$: $t \mapsto (1-t)\boldsymbol{X} + t\boldsymbol{Y}$. For $t = 2$, one gets

$$\boldsymbol{Z} = -\boldsymbol{X} + 2\boldsymbol{Y} = \begin{pmatrix} 1 & -3 \\ -3 & 4 \end{pmatrix}.$$

The smallest eigenvalue of $\boldsymbol{Z}$ is $\frac{1}{2}(5 - 3\sqrt{5}) < 0$. Thus, $\boldsymbol{Z}$ is not SPD.



(a) Ellipses representing the Euclidean t geodesic between $\boldsymbol{X}$ and $\boldsymbol{Y}$.



(b) Evolution of the determinant along the Euclidean geodesic between $\boldsymbol{X}$ and $\boldsymbol{Y}$.

Figure 2.6. Illustration of the swelling effect when using the Euclidean geodesic between two SPD matrices. The matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the ones given in Example 2.10.

Remark 2.11. When the SPD matrix studied is a covariance matrix estimated from an EEG, as described in Section 2.1.4, leaving the cone $\mathcal{P}_d$ has some physical implications. Indeed, if a symmetric matrix is outside of the cone, it has a negative eigenvalue, which means a negative variance (or equivalently, a negative power), which is unfeasible.

This extrapolation problem is not the only issue when using the Euclidean geometry for SPD matrices. Another issue is known as the *swelling effect* and was studied by Arsigny et al. (2006a). This effect translates into an increase of the determinant on the straight line between $\boldsymbol{X}$ and $\boldsymbol{Y}$ when they have the same determinant. Thus, for $t = \frac{1}{2}$, the determinant of the Euclidean mean of $\boldsymbol{X}$ and $\boldsymbol{Y}$, the matrix $\frac{1}{2}(\boldsymbol{X} + \boldsymbol{Y})$, will always be greater than or equal to the geometric mean of the original determinants:

$$\det\left(\frac{\boldsymbol{X}+\boldsymbol{Y}}{2}\right) \geqslant \sqrt{\det(\boldsymbol{X})\det(\boldsymbol{Y})}.$$

This is based on Minkowski's determinant inequality (see Theorem 7.8.21 of Horn and Johnson 2012) and the classical arithmetic-geometric mean inequality. We know that the determinant is linked to volume. In the case of an SPD matrix $\boldsymbol{X}$, as defined by Grötschel et al. (2012, Section 3.1), one can define the associated ellipsoid as a subset $E_{\boldsymbol{X}} \subseteq \mathbb{R}^d$ defined by

$$E_{\boldsymbol{X}} = \{v \in \mathbb{R}^d \mid v^\top \boldsymbol{X}^{-1} v \leqslant 1\}.$$

As explained by Grötschel et al. (2012), it is more convenient to use $\boldsymbol{X}^{-1}$ rather than $\boldsymbol{X}$[6]. Then, the volume of the ellipsoid $E_{\boldsymbol{X}}$ depends only on the determinant of $\boldsymbol{X}$ and the dimension

[6]Moreover, when $\boldsymbol{X}$ is the covariance matrix of a multivariate Gaussian distribution, the density is proportional to $\exp\left(-\frac{1}{2}(v-\mu)^\top \boldsymbol{X}^{-1}(v-\mu)\right)$, thus the confidence region is of the form given by our definition of an ellipsoid.

d of the space:

$$\mathrm{Vol}(E_{\boldsymbol{X}}) = \sqrt{\det \boldsymbol{X}} V_d,$$

where V_d is the volume of the unit ball in $\mathbb{R}^d$. Therefore, the swelling effect shows that when interpolating using the Euclidean straight line, the volume of the interpolated ellipsoid will be greater than the volumes of the original points. This swelling effect phenomenon is illustrated in Figure 2.6 for the same $\boldsymbol{X}$ and $\boldsymbol{Y}$ as in Example 2.10.

Those two downsides make using the Euclidean geometry induced by $\mathcal{S}_d$ on the set of SPD matrices $\mathcal{P}_d$ unreasonable. Fortunately, the set of SPD matrices can be equipped with another non-flat manifold structure, paving the way for a well suited geometry. Some of these non-flat geometries will not suffer from the swelling effect, as we will see in Figure 2.8.

2.2.3 A Riemannian metric for the set of SPD matrices

Using Proposition 2.5, one can endow the set $\mathcal{P}_d$ with a strong structure of a manifold.

Proposition 2.12. *The set of SPD matrices $\mathcal{P}_d$ is a manifold of dimension $d(d+1)/2$. The tangent spaces of $\mathcal{P}_d$ are identified with $\mathcal{S}_d$.*

Proof. This comes from the fact that $\mathcal{P}_d$ is an open set of $\mathcal{S}_d$ (see Proposition 2.5). □

The goal of this section is not to provide a full course on differentiable manifolds but to study the specifics of the manifold $\mathcal{P}_d$ (Pennec 2020; Bhatia 2007). For a general theory on manifolds and Riemannian manifolds, we refer to the two books J. M. Lee (2012) and J. M. Lee (2018) or the great book by Boumal (2023) for a lighter introduction. Having a manifold structure on the set $\mathcal{P}_d$ already gives us a lot of information on this set: locally, this space is smooth, one can study smooth maps defined on $\mathcal{P}_d$ or define tangent spaces and tangent fields. However, if one wants to do any type of learning on this space, as it will be our case in this thesis, one needs a way of comparing points, of measuring geometric quantities such as distances or angles. This is where the structure of a *Riemannian manifold* comes in handy, as it will provide Riemannian metrics that will allow us to do "geometry" in the familiar sense of the word, as Euclid thought of it more than 2000 years ago. The starting point of Riemannian geometry is to define a Riemannian metric on the tangent spaces of the set of interest, here $\mathcal{P}_d$. Many metrics have been studied (Pennec 2020; Arsigny et al. 2006b; Z. Lin 2019; Bhatia et al. 2019) that all fit different purposes. In this thesis, we mainly focus on the *Affine Invariant Riemannian Metric* (Pennec 2020, AIRM), but we also describe in the following the log-Euclidean metric (Arsigny et al. 2006b), the Cholesky (Dryden, Koloydenko, et al. 2009) and log-Cholesky (Z. Lin 2019) metrics and the Bures-Wasserstein metric (Bhatia et al. 2019). Let us start by the main one: the Affine Invariant Riemannian Metric.

2.2.3.1 Affine Invariant Riemannian geometry

The first metric that we describe here and that will be used throughout this manuscript is the *Affine Invariant Riemannian Metric*. This metric was first introduced in Siegel (1943) to introduce an invariant metric on the Siegel half-space $\{\boldsymbol{X} + i\boldsymbol{Y} \mid \boldsymbol{X} \in \mathcal{S}_d, \boldsymbol{Y} \in \mathcal{P}_d\}$. It was then used in the 1980s to model the geometry of the multivariate Gaussian family (Skovgaard 1984). It was then rediscovered in computer vision thanks to Förstner and Moonen (2003). Nowadays, it is one of the most commonly used metrics for SPD matrices (Yger, Berar, et al. 2017; S. Chevallier et al. 2021; Barachant, Bonnet, et al. 2010b). The main reference used to write this section is Pennec (2020), however, most of the results given here can also be found in chapter 6 of Bhatia (2007).

The core idea behind this metric is that a natural transformation on the space of SPD matrices is given by the invertible affine transformation of a feature vector $X \in \mathbb{R}^d \mapsto \boldsymbol{A}X + t$ where $\boldsymbol{A} \in \mathrm{GL}(d)$ and $t \in \mathbb{R}^d$. Then, the covariance matrix of the transformed vector $\boldsymbol{A}X + t$

is congruent to the original one *via* the linear part $\boldsymbol{A}$ of the transformation: $\mathrm{Cov}(\boldsymbol{A}X + t) = \boldsymbol{A}\,\mathrm{Cov}(X)\boldsymbol{A}^\top$. Thus, choosing an affine invariant metric will imply that the geometry of a dataset will not change if we apply the same affine transformation to the whole dataset. We motivated this from a BCI point of view in Section 2.1.4.3.

Let us denote $\star$ the action by congruence of $\mathrm{GL}(d)$ on $\mathcal{P}_d$: $\boldsymbol{A} \star \boldsymbol{X} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^\top$. We are thus seeking a metric that is invariant to $\star$, meaning that

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \langle \boldsymbol{A} \star \boldsymbol{U}, \boldsymbol{A} \star \boldsymbol{V} \rangle_{\boldsymbol{A}\star\boldsymbol{X}}, \tag{2.2}$$

where $\boldsymbol{X} \in \mathcal{P}_d, \boldsymbol{U}, \boldsymbol{V} \in T_{\boldsymbol{X}}\mathcal{P}_d$ and $\boldsymbol{A} \in \mathrm{GL}(d)$. For this, we start by choosing one of the simplest inner products on the tangent space at the identity (that is identified with $\mathcal{S}_d$), namely the Frobenius inner product. Then, using $\boldsymbol{A} = \boldsymbol{X}^{-1/2}$ (see Theorem 2.4) and the affine invariance given in Equation (2.2), we can deduce the metric at $\boldsymbol{X}$ from the metric at the identity.

Definition 2.13 (Affine Invariant Riemannian Metric)**.** Let $\boldsymbol{X} \in \mathcal{P}_d$. The *Affine Invariant Riemannian Metric* (AIRM) is defined on the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ at $\boldsymbol{X}$ as follows:

$$\forall \boldsymbol{U}, \boldsymbol{V} \in T_{\boldsymbol{X}}\mathcal{P}_d,\ \langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \mathrm{tr}(\boldsymbol{X}^{-1}\boldsymbol{U}\boldsymbol{X}^{-1}\boldsymbol{V}).$$

This application $\boldsymbol{X} \mapsto \langle \cdot, \cdot \rangle_{\boldsymbol{X}}$ varies smoothly from point to point,[7] thus, $(\mathcal{P}_d, \langle \cdot, \cdot \rangle_{\boldsymbol{X}})$ is a Riemannian manifold. Using this metric and thanks to Proposition 2.9, we have an important result on the structure of the Riemannian manifold $\mathcal{P}_d$ equipped with the AIRM metric: it is a *non-compact Riemannian symmetric space*.

Once we have endowed the space $\mathcal{P}_d$ with the AIRM metric from Definition 2.13, we are able to compute the geodesics between two points $\boldsymbol{X}$ and $\boldsymbol{Y}$ of $\mathcal{P}_d$. This geodesic is the shortest path between $\boldsymbol{X}$ and $\boldsymbol{Y}$ under the AIRM geometry.

Proposition 2.14. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$. The unique geodesic connecting $\boldsymbol{X}$ and $\boldsymbol{Y}$ under the AIRM geometry is given by:*

$$\forall t \in [0,1],\ \gamma(t) = \boldsymbol{X}^{1/2}\left(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2}\right)^t \boldsymbol{X}^{1/2}.$$

Let us make two remarks about this geodesic.

Remark 2.15. We are also able to compute the geodesic γ starting at $\boldsymbol{X} \in \mathcal{P}_d$ and with initial velocity $\boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d$:

$$\forall t \in [0,1],\ \gamma(t) = \boldsymbol{X}^{1/2}\exp\left(t\boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2}\right)\boldsymbol{X}^{1/2}. \tag{2.3}$$

Remark 2.16. Recall from Section 2.2.2 that the *swelling effect*, an artificial increase of the determinant along the straight line between two SPD matrices of equal determinant, was one of the arguments against the Euclidean geometry on $\mathcal{P}_d$. We now show that this artifact disappears under the AIRM geodesic of Proposition 2.14.
We first establish a useful identity: for any diagonalizable matrix $\boldsymbol{Z}$ with eigenvalues $\lambda_1, \ldots, \lambda_d$,

$$\det(\boldsymbol{Z}^t) = \det\big(\exp(t\log \boldsymbol{Z})\big) = \exp\big(t\ \mathrm{tr}(\log \boldsymbol{Z})\big) = \exp\Big(t\sum_{i=1}^{d}\log\lambda_i\Big) = \prod_{i=1}^{d}\lambda_i^t = \det(\boldsymbol{Z})^t,$$

where the second equality uses $\det(\exp(\boldsymbol{A})) = \exp(\mathrm{tr}(\boldsymbol{A}))$. Now let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$ with

[7] To be fully precise, for all smooth vector fields V, W on $\mathcal{P}_d$, the function $\boldsymbol{X} \mapsto \langle V(\boldsymbol{X}), W(\boldsymbol{X}) \rangle_{\boldsymbol{X}}$ is smooth from $\mathcal{P}_d$ to $\mathbb{R}$.

$\det(\boldsymbol{X}) = \det(\boldsymbol{Y})$, and let γ be the AIRM geodesic between them. For $t \in [0,1]$,

$$\begin{aligned}\det(\gamma(t)) &= \det\Big(\boldsymbol{X}^{1/2}(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2})^t\boldsymbol{X}^{1/2}\Big)\\ &= \det(\boldsymbol{X})\det(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2})^t\\ &= \det(\boldsymbol{X})\det(\boldsymbol{X})^{-t}\det(\boldsymbol{Y})^t\\ &= \det(\boldsymbol{X})^{1-t}\det(\boldsymbol{Y})^t,\end{aligned}$$

where the second line uses the identity above together with multiplicativity of the determinant. Since $\det(\boldsymbol{X}) = \det(\boldsymbol{Y})$, the determinant is constant along γ: there is no swelling effect.

Looking at Equation (2.3), one can see that every geodesic in the AIRM geometry can be extended to a geodesic defined on the whole real line. Thus, the manifold $\mathcal{P}_d$ equipped with the AIRM geometry is *geodesically and metrically complete*. As discussed in Pennec (2020), "From a computational point of view, the geodesic completeness of the affine-invariant structure ensures that any finite geodesic walking scheme will remain in the space of SPD matrices, contrarily to the Euclidean case". Moreover, using Theorem 10.9 of Boumal (2023), we are able to integrate the length of the geodesic in order to compute the AIRM distance between two SPD matrices.

Corollary 2.17. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$. The AIRM distance δ^{AIRM} between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is given by:*

$$\delta^{AIRM}(\boldsymbol{X},\boldsymbol{Y}) = \|\log(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2})\|_F = \left(\sum_{i=1}^{d}\log^2\lambda_i(\boldsymbol{X}^{-1}\boldsymbol{Y})\right)^{1/2}, \tag{2.4}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\lambda_i(\boldsymbol{X}^{-1}\boldsymbol{Y})$ are the eigenvalues of $\boldsymbol{X}^{-1}\boldsymbol{Y}$[a].

[a] Or equivalently, the solutions λ of the generalized eigenvalue problem $\boldsymbol{Y}\mathbf{v} = \lambda\boldsymbol{X}\mathbf{v}$.

Remark 2.18. The AIRM metric defined in Definition 2.13 is also deeply rooted in information geometry as it is the Fisher–Rao metric of the zero-mean multivariate Gaussian model (up to a factor 2) (Skovgaard 1984). This is thus the natural metric that information geometry associates to this statistical model $\{\mathcal{N}(0,\boldsymbol{X}) \mid \boldsymbol{X} \in \mathcal{P}_d\}$. In particular, the geodesic distance $\delta(\boldsymbol{X},\boldsymbol{Y})$ can be interpreted as the integrated Fisher information along the path of Gaussian distributions connecting $\mathcal{N}(0,\boldsymbol{X})$ to $\mathcal{N}(0,\boldsymbol{Y})$. This statistical grounding justifies the use of the AIRM in settings where SPD matrices arise as sample covariances of a Gaussian process, as is standard in EEG-based BCI pipelines, see Section 2.1.4.2.

On any Riemannian manifold, one can define the *exponential mapping*[8] that maps the tangent space at $\boldsymbol{X}$ to a neighborhood of $\boldsymbol{X}$ on the manifold. In full generality, this mapping is a local diffeomorphism. However, in the case of the AIRM geometry on $\mathcal{P}_d$, this mapping is a *global* diffeomorphism. One can thus define its global inverse mapping, called the *logarithmic mapping*.

Proposition 2.19. *The Riemannian exponential and logarithm mappings for the manifold $\mathcal{P}_d$ of SPD matrices endowed with the AIRM metric defined in Definition 2.13 are global diffeomorphisms and are given by, for $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$ and $\boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d$,*

$$\begin{aligned}\mathrm{Exp}_{\boldsymbol{X}}(\boldsymbol{U}) &= \boldsymbol{X}^{1/2}\exp(\boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2})\boldsymbol{X}^{1/2},\\ \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) &= \boldsymbol{X}^{1/2}\log(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2})\boldsymbol{X}^{1/2}.\end{aligned}$$

[8] This mapping is called the exponential because it corresponds to the usual exponential in some matrix groups.

Another useful and important mapping that we will need is the *parallel transport* that enables us to transport a tangent vector $\boldsymbol{U}$ from the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ at $\boldsymbol{X}$ to another tangent space $T_{\boldsymbol{Y}}\mathcal{P}_d$ at $\boldsymbol{Y}$.

Proposition 2.20. *The parallel transport $\Gamma_{\boldsymbol{X}\to\boldsymbol{Y}}$ from $\boldsymbol{X} \in \mathcal{P}_d$ to $\boldsymbol{Y} \in \mathcal{P}_d$ according to the AIRM geometry is given by, for $\boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d$:*

$$\Gamma_{\boldsymbol{X}\to\boldsymbol{Y}}(\boldsymbol{U}) = (\boldsymbol{Y}\boldsymbol{X}^{-1})^{1/2}\boldsymbol{U}(\boldsymbol{X}^{-1}\boldsymbol{Y})^{1/2}.$$

Proof. See Sra and R. Hosseini (2015) for a proof. □

An important characteristic of a Riemannian manifold is its curvature. The idea behind curvature is to measure how much a multidimensional space differs from a flat Euclidean space. Intuitively, when the sectional curvature is equal to zero, the space is (locally) flat, when it is positive, it (locally) looks like a sphere, thus nearby geodesics will converge, and when the sectional curvature is negative, the space (locally) looks like a saddle, thus the geodesics diverge. We refer to Chapter 7 of J. M. Lee (2018) for a proper study of the curvature. In the case of the manifold $\mathcal{P}_d$ equipped with the AIRM geometry, we are able to give bounds on the sectional curvature.

Proposition 2.21. *The sectional curvature of $\mathcal{P}_d$ is in $\left[-\frac{1}{2}, 0\right]$ and is not constant.*

Proof. The proof can be found in the Appendix I of Criscitiello and Boumal (2023) or Section 3.3.4 of Pennec (2020). □

Thus, for $\mathcal{P}_d$ equipped with AIRM, geodesics with a small initial angular separation will drift apart exponentially. This also means that volumes in this space grow faster than in a Euclidean space and that geodesic triangles are "thin", the sum of their angles is less than π. The previous proposition is the last ingredient needed to state that $\mathcal{P}_d$, equipped with the AIRM metric, is a *Hadamard manifold*, a complete, simply connected Riemannian manifold of non-positive sectional curvature everywhere.

In Chapter 3 and Chapter 4, we will study probability distributions defined on the manifold $\mathcal{P}_d$ equipped with the AIRM metric. For this, we will need to integrate over the manifold $\mathcal{P}_d$. A key element allowing us to integrate is the *Riemannian volume form*. We now give its expression.

Proposition 2.22. *The Riemannian volume form of $\mathcal{P}_d$ equipped with the AIRM metric is given by*

$$\mathrm{d}\,\mathrm{vol}(\boldsymbol{X}) = 2^{\frac{d(d-1)}{4}} \det(\boldsymbol{X})^{-\frac{d+1}{2}} \prod_{i\leqslant j} \mathrm{d}X_{ij},$$

where $\boldsymbol{X} = (X_{ij})_{1\leqslant i,j\leqslant d} \in \mathcal{P}_d$ and $\mathrm{d}X_{ij}$ are the Lebesgue measures on $\mathbb{R}$.

The only reference we found for the expression of this volume form is in Terras (1988, Section 4.1.3), but we never found a proof of this statement. An *ad hoc* proof is provided in the appendix in Appendix A.

Remark 2.23. As stated at the beginning of this section, the AIRM defined in Definition 2.13 is affine invariant, and it has been constructed by choosing the Frobenius metric on $T_{I_d}\mathcal{P}_d$, the tangent space at I_d. However, choosing another inner product for $T_{I_d}\mathcal{P}_d$ could lead to another affine invariant metric. Pennec showed (see Section 3.3.3 of Pennec 2020) that all metrics on $\mathcal{P}_d$ that are invariant under the $\mathrm{GL}(d)$ action $\boldsymbol{A} \star \boldsymbol{X} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^\top$, are given by the one-parameter family (up to a global scaling factor):

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \mathrm{tr}(\boldsymbol{X}^{-1}\boldsymbol{U}\boldsymbol{X}^{-1}\boldsymbol{V}) + \beta\,\mathrm{tr}(\boldsymbol{U}\boldsymbol{X}^{-1})\,\mathrm{tr}(\boldsymbol{V}\boldsymbol{X}^{-1}),$$

where $\beta > -\frac{1}{d}$. One retrieves the classical AIRM metric for $\beta = 0$. An interesting fact about this family of affine invariant metrics is that they have the same Levi-Civita connection (see Chapter 5 of J. M. Lee 2018, for the definition of the Levi-Civita connection), thus they share the same geodesics and Riemannian exponential and logarithmic mappings. However, the Riemannian distance is then:

$$\delta_\beta^{\text{AIRM}}(\boldsymbol{X}, \boldsymbol{Y})^2 = \| \log(\boldsymbol{X}^{-1/2} \boldsymbol{Y} \boldsymbol{X}^{-1/2}) \|_F^2 + \beta \log^2 \det(\boldsymbol{X}^{-1} \boldsymbol{Y}).$$

Remark 2.24. The AIRM geometry has a lot of nice mathematical properties: it is an affine invariant metric that gives $\mathcal{P}_d$ a Hadamard and non-compact symmetric space structure, and all the Riemannian operations can be explicitly computed, but it also has some drawbacks in practice. Let us mention some of those drawbacks:

- **Computationally costly:** Every operation performed on the set $\mathcal{P}_d$ using the AIRM geometry (distance, exponential or logarithmic mapping, etc.) relies on computing the square root or the inverse square root of an SPD matrix, which, in practice, is done by computing the eigenvalues and eigenvectors of the matrix. For a $d \times d$ matrix, a full eigendecomposition costs $O(d^3)$ as it relies on a QR decomposition (we refer to Golub and Van Loan 2013, for more insights on the QR decomposition). Modern Python libraries such as NumPy (Harris et al. 2020) or SciPy (Virtanen et al. 2020) rely on highly optimized LAPACK routines (typically written in Fortran or C). However, computing the eigendecomposition of a lot of high-dimensional matrices can be a real bottleneck in practice.
- **No closed form for the Fréchet mean:** Given a set of N SPD matrices $\boldsymbol{X}_1, ... \boldsymbol{X}_N \in \mathcal{P}_d$, a first order statistic of this set is the Fréchet mean $\mathfrak{G}(\boldsymbol{X}_1, ..., \boldsymbol{X}_N)$ (Fréchet 1948) defined as:

  $$\mathfrak{G}(\boldsymbol{X}_1, ..., \boldsymbol{X}_N) = \underset{\boldsymbol{X} \in \mathcal{P}_d}{\arg\min} \sum_{i=1}^{N} \delta(\boldsymbol{X}, \boldsymbol{X}_i)^2.$$

  When equipped with the AIRM, the Fréchet mean has a unique solution (Karcher 1977), as $\mathcal{P}_d$ is of non-positive sectional curvature (see Proposition 2.21). However, this Fréchet mean has no closed-form formula and computing it relies on a Riemannian gradient descent (Moakher 2005).

2.2.3.2 The Log-Euclidean geometry

In order to build another Riemannian metric for the set of SPD matrices $\mathcal{P}_d$, Vincent Arsigny, Pierre Fillard, Xavier Pennec and Nicolas Ayache used the fact that the matrix exponential is a diffeomorphism between $\mathcal{S}_d$ and $\mathcal{P}_d$ (see Theorem 2.6) to transfer the Euclidean structure from $\mathcal{S}_d$ to $\mathcal{P}_d$ using a *pullback metric* in Arsigny et al. (2006a) and Arsigny et al. (2006b). One can thus transform the classical Frobenius metric on $\mathcal{S}_d$ into a Riemannian metric on $\mathcal{P}_d$, called the *log-Euclidean metric*.

Definition 2.25 (Log-Euclidean metric)**.** Let $\boldsymbol{X} \in \mathcal{P}_d$. The *log-Euclidean metric* is defined on the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ as follows:

$$\forall \boldsymbol{U}, \boldsymbol{V} \in T_{\boldsymbol{X}}\mathcal{P}_d, \ \langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \ \langle \mathrm{d}\log(\boldsymbol{X}) \cdot \boldsymbol{U}, \mathrm{d}\log(\boldsymbol{X}) \cdot \boldsymbol{V} \rangle_F,$$

where $\mathrm{d}\log(\boldsymbol{X})$ is the differential of log at $\boldsymbol{X}$.

This definition uses the differential of log that is discussed in Section 3.2.2 of Pennec (2020). From this definition, one has that the set of SPD matrices $\mathcal{P}_d$ is a flat space when endowed with the log-Euclidean metric. Moreover, one is able to compute the corresponding Riemannian exponential and logarithmic mappings, geodesics, as well as the log-Euclidean

distance:

Proposition 2.26. *Let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$ and $\boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d$. The exponential and logarithmic mappings are given by*

$$\begin{aligned}\mathrm{Exp}_{\boldsymbol{X}}(\boldsymbol{U}) &= \exp(\log \boldsymbol{X} + \mathrm{d}\log(\boldsymbol{X}) \cdot \boldsymbol{U}),\\ \mathrm{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) &= \mathrm{d}\exp(\log \boldsymbol{X}) \cdot (\log \boldsymbol{X} - \log \boldsymbol{Y}).\end{aligned}$$

Moreover, the geodesic between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is given by, at $t \in [0,1]$:

$$\gamma(t) = \exp((1-t)\log \boldsymbol{X} + t \log \boldsymbol{Y}).$$

Finally, the log-Euclidean distance δ^{LE} between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is given by:

$$\delta^{LE}(\boldsymbol{X}, \boldsymbol{Y}) = \| \log \boldsymbol{X} - \log \boldsymbol{Y} \|_F.$$

Proof. These statements are proven in Arsigny et al. (2006b). □

In practice, the log-Euclidean framework consists in taking the matrix logarithm of the SPD matrices, computing distances or geodesics as usual in the Euclidean space of symmetric matrices, and coming back at the end to the SPD matrix space using the exponential (Arsigny et al. 2006a; Arsigny et al. 2006b). The vector space structure gives a closed-form expression of some important quantities. An important example is the log-Euclidean mean.

Proposition 2.27 (Log-Euclidean Mean)**.** *Let $\boldsymbol{X}_1, ..., \boldsymbol{X}_N \in \mathcal{P}_d$. The log-Euclidean mean has a closed form formula given by:*

$$\exp\left(\frac{1}{N}\sum_{i=1}^{N} \log(\boldsymbol{X}_i)\right).$$

Pennec (2020) states that "one can conclude that the log-Euclidean framework is a good first order approximation of the affine-invariant computations."

Remark 2.28. In the definition of the log-Euclidean metric given in Definition 2.25, the pullback chosen is the classical matrix logarithm. This is equivalent to sending the points to $T_{I_d}\mathcal{P}_d$, the tangent space at I_d, using the AIRM geometry (as the Riemannian logarithm at I_d for the AIRM geometry is simply the matrix logarithm, see Proposition 2.19). However, if another reference point $\boldsymbol{Z}$ is more central than the identity for the data, one can choose to project the points to the tangent space $T_{\boldsymbol{Z}}\mathcal{P}_d$ at $\boldsymbol{Z}$, thus consider the log-Euclidean structure induced by $\boldsymbol{X} \mapsto \boldsymbol{Z}^{1/2}\log(\boldsymbol{Z}^{-1/2}\boldsymbol{X}\boldsymbol{Z}^{-1/2})\boldsymbol{Z}^{1/2}$ instead of $\boldsymbol{X} \mapsto \log(\boldsymbol{X})$. The log-Euclidean distance then becomes:

$$\delta_{\boldsymbol{Z}}^{\mathrm{LE}}(\boldsymbol{X}, \boldsymbol{Y}) = \| \log(\boldsymbol{Z}^{-1/2}\boldsymbol{X}\boldsymbol{Z}^{-1/2}) - \log(\boldsymbol{Z}^{-1/2}\boldsymbol{Y}\boldsymbol{Z}^{-1/2}) \|_F.$$

2.2.3.3 The Cholesky and log-Cholesky geometries

An important linear algebra and numerical analysis statement on SPD matrices is the Cholesky decomposition (see Theorem 4.2.7 of Golub and Van Loan 2013) that we recall here:

Theorem 2.29 (Cholesky Factorization)**.** *If $\boldsymbol{X} \in \mathbb{R}^{d\times d}$ is symmetric positive definite, then there exists a unique lower triangular $\boldsymbol{L} \in \mathbb{R}^{d\times d}$ with positive diagonal entries such that*

$$\boldsymbol{X} = \boldsymbol{L}\boldsymbol{L}^{\top}. \tag{2.5}$$

$\boldsymbol{L}$ is called the Cholesky factor *of $\boldsymbol{X}$ and is denoted $\mathcal{L}(\boldsymbol{X})$.*

The mapping $\mathcal{L}$ from $\mathcal{P}_d$ to $\mathfrak{L}^+$, the space of lower triangular $\boldsymbol{L} \in \mathbb{R}^{d\times d}$ with positive diagonal entries, is bijective. In fact, Z. Lin (2019) even showed that it was a diffeomorphism between the smooth manifolds $\mathfrak{L}^+$ (identified with $\mathbb{R}^{d(d+1)/2}$) and $\mathcal{P}_d$.

In Z. Wang et al. (2004), the authors parametrized SPD matrices by their Cholesky decomposition when working with DTI fields. A first way of using the Cholesky factorization to build a Riemannian metric on $\mathcal{P}_d$ is to endow the manifold $\mathfrak{L}^+$ with a Euclidean geometry and transport it to $\mathcal{P}_d$ by the product given in Equation (2.5). According to Dryden, Koloydenko, et al. (2009), this leads to a flat geometry on $\mathcal{P}_d$ and the distance is given by:

$$\delta^{\mathrm{C}}(\boldsymbol{X}, \boldsymbol{Y}) = \|\mathcal{L}(\boldsymbol{X}) - \mathcal{L}(\boldsymbol{Y})\|_2.$$

We will call this geometry the *Cholesky geometry* in the following. As this geometry arises from a pullback metric, computing the mean can also be done in closed form, as is the case for the log-Euclidean metric (see Proposition 2.27).

Proposition 2.30. *Let $\boldsymbol{X}_1, ..., \boldsymbol{X}_N \in \mathcal{P}_d$. The Cholesky mean $\boldsymbol{M}$ is $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top$, where*

$$\boldsymbol{L} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{X}_i).$$

Another way of using the Cholesky decomposition to build a geometry on $\mathcal{P}_d$ is to proceed similarly as for the Log-Euclidean metric described above, that is to construct a group structure and a metric on the manifold $\mathfrak{L}^+$ and to push them onto $\mathcal{P}_d$ via the Cholesky map. This is what Z. Lin (2019) did. For this, the authors split a lower triangular matrix $\boldsymbol{L} \in \mathfrak{L}^+$ into two parts: the diagonal $\mathbb{D}(\boldsymbol{L})$ (which is strictly positive) and the lower triangular part below the diagonal $\lfloor \boldsymbol{L} \rfloor$. Then, $\boldsymbol{L} = \mathbb{D}(\boldsymbol{L}) + \lfloor \boldsymbol{L} \rfloor$. Using this split, one can introduce an inner product on the tangent space $T_{\boldsymbol{L}}\mathfrak{L}^+$ at $\boldsymbol{L} \in \mathfrak{L}^+$, which is identified with $\mathfrak{L}$, the space of lower triangular matrices. This inner product is:

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{L}}^{\mathfrak{L}^+} = \langle \lfloor \boldsymbol{U} \rfloor, \lfloor \boldsymbol{V} \rfloor \rangle_F + \langle \mathbb{D}(\boldsymbol{L})^{-1}\mathbb{D}(\boldsymbol{U}), \mathbb{D}(\boldsymbol{L})^{-1}\mathbb{D}(\boldsymbol{V}) \rangle_F.$$

Equipped with this metric, the manifold $\mathfrak{L}^+$ is a Riemannian manifold, and we can now transfer this structure on the set of SPD matrices *via* the pullback $\mathcal{L}$, the Cholesky decomposition. This leads to the *log-Cholesky geometry* and the corresponding inner product is then

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \langle \mathrm{d}\mathcal{L}(\boldsymbol{X}) \cdot \boldsymbol{U}, \mathrm{d}\mathcal{L}(\boldsymbol{X}) \cdot \boldsymbol{V} \rangle_{\mathcal{L}(\boldsymbol{X})}^{\mathfrak{L}^+}.$$

One can compute the differential of $\mathcal{L}$ at $\boldsymbol{X}$ using the following formula:

$$\mathrm{d}\mathcal{L}(\boldsymbol{X}) \cdot \boldsymbol{U} = \mathcal{L}(\boldsymbol{X}) \left(\mathcal{L}(\boldsymbol{X})^{-1} \boldsymbol{U} \mathcal{L}(\boldsymbol{X})^{-\top} \right)_{\frac{1}{2}},$$

where $\boldsymbol{A}_{\frac{1}{2}} = \frac{1}{2}\mathbb{D}(\boldsymbol{A}) + \lfloor \boldsymbol{A} \rfloor$. This geometry is flat, and one can compute the distance between two SPD matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ as follows:

$$\delta^{\mathrm{LC}}(\boldsymbol{X}, \boldsymbol{Y}) = \left(\|\lfloor \mathcal{L}(\boldsymbol{X}) \rfloor - \lfloor \mathcal{L}(\boldsymbol{Y}) \rfloor\|_F^2 + \|\log(\mathbb{D}(\mathcal{L}(\boldsymbol{X}))) - \log(\mathbb{D}(\mathcal{L}(\boldsymbol{Y})))\|_F^2 \right)^{1/2}.$$

We refer to Table 2 of Z. Lin (2019) for all the expressions of classical Riemannian operations using this geometry. In Z. Lin (2019), the author argues that this log-Cholesky geometry can be useful when dealing with very large datasets as it is computationally attractive. They claim that for the computation of parallel transport for example, the log-Cholesky "could be approximately 45 times faster than the log-Euclidean metric and 4 times faster than the affine-invariant one." Moreover, it has a closed-form for Fréchet means and does not suffer from the swelling effect.

Metric	Curvature	Closed-form mean	Complexity	No Swelling effect
AIRM	$\leqslant 0$	✗	$\mathcal{O}(N^2d^3)$	✓
Log-Euclidean	0	✓	$\mathcal{O}(Nd^3)$	✓
Cholesky	0	✓	$\mathcal{O}(Nd^3)$	✗
Log-Cholesky	0	✓	$\mathcal{O}(Nd^3)$	✓
Bures-Wasserstein	$\geqslant 0$	✗	$\mathcal{O}(N^2d^3)$	✗

Table 2.2. Comparison of the main Riemannian metrics described here for the manifold of SPD matrices. *Curvature*: what is the sign of the sectional curvature. *Complexity*: complexity of `pairwise_distance` for a set of N matrices of dimension d.

2.2.3.4 The Bures-Wasserstein geometry

The last geometry we will investigate in this chapter is the Bures-Wasserstein geometry. This distance originates from optimal transport. We refer to Villani (2003) and to Peyré and Cuturi (2019) for an introduction to optimal transport. The Bures-Wasserstein distance is exactly the 2-Wasserstein distance between two centered Gaussian measures $\mathcal{N}(0, \boldsymbol{X})$ and $\mathcal{N}(0, \boldsymbol{Y})$ where $\boldsymbol{X}$ and $\boldsymbol{Y}$ represent the dispersion (or covariance) matrices of the measures (see Remark 2.31 of Peyré and Cuturi 2019, for a proof). Its name also comes from the Bures distance used in quantum information geometry (Bures 1969). This distance function is given as

$$\delta^{\mathrm{BW}}(\boldsymbol{X}, \boldsymbol{Y}) = \left(\operatorname{tr} \boldsymbol{X} + \operatorname{tr} \boldsymbol{Y} - 2 \operatorname{tr}\left[(\boldsymbol{X}^{1/2}\boldsymbol{Y}\boldsymbol{X}^{1/2})^{1/2}\right]\right)^{1/2}.$$

This distance also has a Riemannian flavor as it is the quotient Riemannian distance induced by the submersion $\pi\colon \boldsymbol{X} \in \mathrm{GL}(d) \mapsto \boldsymbol{X}\boldsymbol{X}^\top \in \mathcal{P}_d$ and the Frobenius metric on $\mathrm{GL}(d)$. It comes from the quotient structure of $\mathcal{P}_d$ given in Proposition 2.9. The associated quotient Riemannian metric is given by

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \operatorname{tr}\left(\mathcal{S}_{\boldsymbol{X}}(\boldsymbol{U})\boldsymbol{X}\mathcal{S}_{\boldsymbol{X}}(\boldsymbol{V})\right),$$

where $\mathcal{S}_{\boldsymbol{X}}(\boldsymbol{U}) \in \mathcal{S}_d$ is the unique solution to the Sylvester equation $\boldsymbol{X}\mathcal{S}_{\boldsymbol{X}}(\boldsymbol{U}) + \mathcal{S}_{\boldsymbol{X}}(\boldsymbol{U})\boldsymbol{X} = \boldsymbol{U}$ (Sylvester 1884). In Theorem 5 of Bhatia et al. (2019), the authors give an explicit formula for the metric above. Both Bhatia et al. (2019) and Malagò et al. (2018) studied more deeply this geometry. Unlike the AIRM metric given in Definition 2.13, the Bures-Wasserstein metric is only invariant to orthogonal transformations and not fully congruence-invariant like the AIRM. Another major difference with the AIRM geometry is that, once equipped with the Bures-Wasserstein metric, the space $\mathcal{P}_d$ is of non-negative sectional curvature (Takatsu 2011). Moreover, the Wasserstein mean can be iteratively computed as in Zheng et al. (2023). Finally, Thanwerdas and Pennec (2023a) showed that the Bures-Wasserstein geometry extends naturally to positive semidefinite (rank-deficient) matrices.

2.2.3.5 Comparing the different geometries

Let us now compare the five different geometries for $\mathcal{P}_d$ that we have introduced above: the AIRM, the log-Euclidean, the Cholesky, the log-Cholesky and the Bures-Wasserstein. The important differences between them are summarized in Table 2.2. A first important point is the computational efficiency. Let us consider a classical scenario that will appear in the applications of this work: computing a distance matrix for N SPD matrices $\boldsymbol{X}_1, ..., \boldsymbol{X}_N \in \mathcal{P}_d$, that is computing the matrix where the (i, j)-th entry is $\delta(\boldsymbol{X}_i, \boldsymbol{X}_j)$. This can be done using the `pairwise_distance` function from pyRiemann (Barachant, Barthélemy, et al. 2026).

- **AIRM:** For the AIRM geometry, one can precompute $\boldsymbol{X}_i^{-1/2}$ for all N matrices in $\mathcal{O}(Nd^3)$, and then solve one eigenvalue problem per pair, each costing $\mathcal{O}(d^3)$, for a total complexity of $\mathcal{O}(N^2d^3)$.

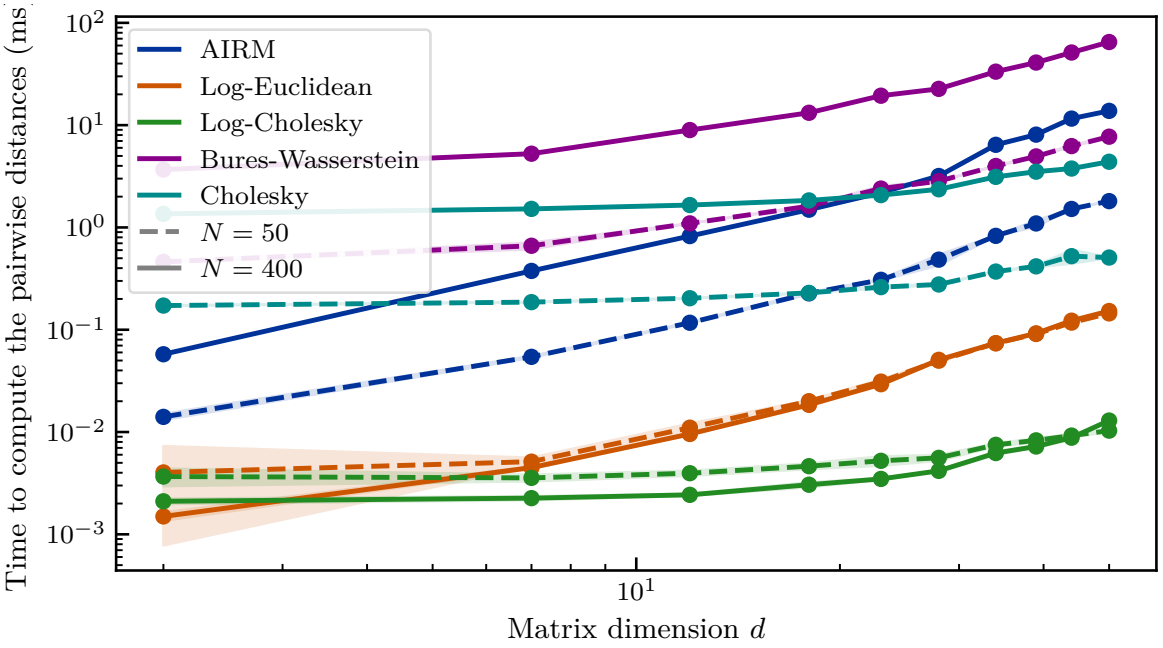


Figure 2.7. Time per matrix for `pairwise_distance` as a function of the dimension d, for two values of N ($N = 50$: dashed, $N = 400$: solid). Each measurement is repeated 5 times; shaded bands show ± 1 standard deviation. Metrics whose two curves overlap are N-independent ($O(Nd^3)$); metrics whose curves are separated by a factor $400/50 = 8$ scale linearly with N ($O(N^2d^3)$).

- **Log-Euclidean:** For the log-Euclidean geometry, one first computes the N matrix logarithms via eigendecomposition in $\mathcal{O}(Nd^3)$, and then evaluates pairwise Frobenius norms in $\mathcal{O}(N^2d^2)$. The dominant cost is thus $\mathcal{O}(Nd^3)$.
- **Cholesky:** One first computes the N Cholesky factors in $\mathcal{O}(Nd^3)$, and then evaluates pairwise Frobenius norms of triangular matrices in $\mathcal{O}(N^2d^2)$. The dominant cost is thus $\mathcal{O}(Nd^3)$.
- **Log-Cholesky:** One first computes the N Cholesky factors in $\mathcal{O}(Nd^3)$, and then evaluates pairwise distances on triangular matrices in $\mathcal{O}(N^2d^2)$. The dominant cost is thus $\mathcal{O}(Nd^3)$.
- **Bures-Wasserstein:** One first precomputes $\boldsymbol{X}_i^{1/2}$ for all N matrices in $\mathcal{O}(Nd^3)$, and then computes $\left(\boldsymbol{X}_i^{1/2}\boldsymbol{Y}_j\boldsymbol{X}_i^{1/2}\right)^{1/2}$ via one eigendecomposition per pair $(\boldsymbol{X}_i, \boldsymbol{Y}_j)$, costing $\mathcal{O}(d^3)$ each. The total complexity is thus $\mathcal{O}(N^2d^3)$.

In order to check numerically those complexities, we conducted a small numerical experiment. We generated $N = 200$ random SPD matrices[9] for ten values of the dimension d ranging from 2 to 50, and measured the wall-clock time[10] required to compute the full pairwise distance matrix under each of the five metrics. Each measurement was repeated five times; the mean and standard deviation are shown in Figure 2.7. The results confirm the theoretical analysis: the AIRM and Bures-Wasserstein metrics, which require one eigendecomposition *per pair*, exhibit a steep $\mathcal{O}(N^2d^3)$ scaling and are substantially slower at large d. Moreover, going from $N = 50$ matrices to $N = 400$ matrices significantly slows the computations. On the other hand, the log-Euclidean and log-Cholesky metrics, which precompute the N decompositions once and then evaluate cheap pairwise norms, grow much more slowly and remain competitive even at $d = 50$. The effect of N for those metrics is less visible. For the Cholesky metric, we

[9] Each matrix is constructed as $\boldsymbol{Q}\,\mathrm{diag}(\lambda)\,\boldsymbol{Q}^\top$, where the eigenvectors $\boldsymbol{Q} \in \mathrm{O}(d)$ are drawn from the Haar measure by QR-decomposing a matrix with i.i.d. $\mathcal{N}(0,1)$ entries (Proposition 7.3 of Eaton 2007), and the eigenvalues λ_i are sampled independently and uniformly from $[0.5, 2.0]$.

[10] On a MacBook Pro M3 equipped with 36 GB of memory.

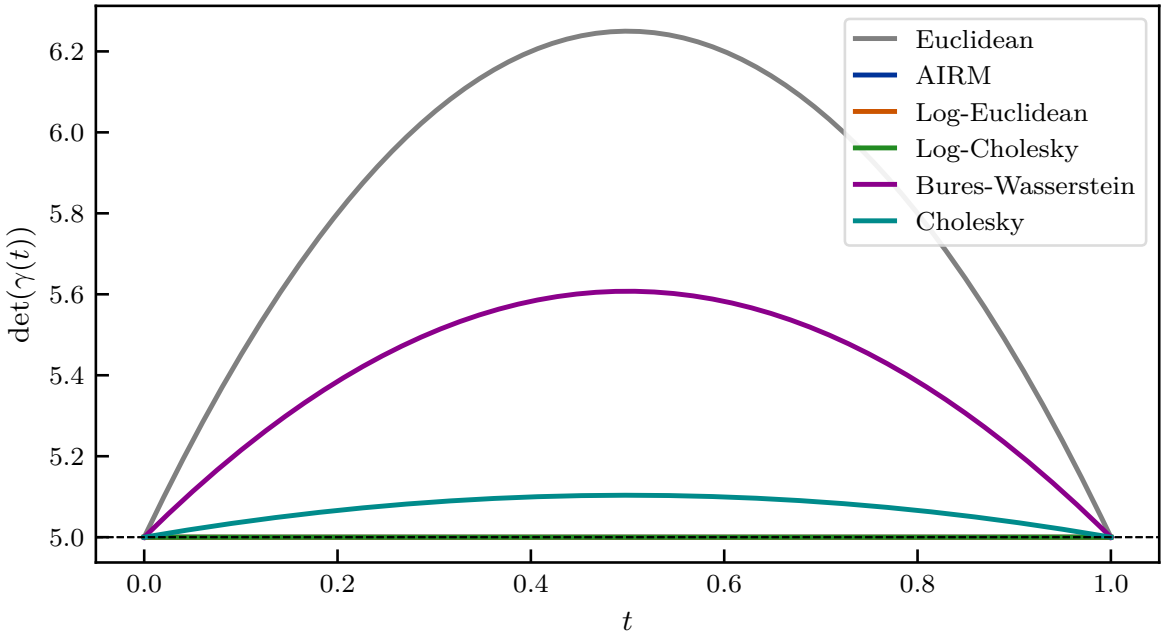


Figure 2.8. Comparison of the determinant along the geodesic γ of the different geometries presented above. If the line is not straight, then the corresponding geometry suffers from the swelling effect.

must say that its implementation for `pairwise_distance` in pyRiemann is not optimal, as it performs a double `for` loop, without precomputing the Cholesky decomposition of each matrix. This explains the slow computational time for this metric.

One of the flaws of the Euclidean geometry discussed in Section 2.2.2 was the *swelling effect*, an increase of the determinant along the Euclidean geodesic. Let us look (numerically) at which of the above geometries, among the AIRM, the log-Euclidean, the Cholesky, the log-Cholesky and the Bures-Wasserstein, still exhibit a swelling effect. Recall that we have proven that the AIRM does not suffer from this effect in Remark 2.16. For the other geometries, let us consider the same 2×2 SPD matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ as in Example 2.10, and we plot the determinant along the geodesic between $\boldsymbol{X}$ and $\boldsymbol{Y}$ according to the different geometries in Figure 2.8. From this plot, we get that the AIRM, the log-Euclidean as well as the log-Cholesky do not suffer from the swelling effect, while the Euclidean, the Cholesky and the Bures-Wasserstein exhibit an increase of the determinant on the geodesic between $\boldsymbol{X}$ and $\boldsymbol{Y}$.

2.2.3.6 Other geometries on $\mathcal{P}_d$

Other geometries have been proposed for the manifold of SPD matrices. Let us give the expression of some of the distances that are used for SPD matrices. For the following, we consider $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$.

- **Power Euclidean metrics:** A broad family of metrics on $\mathcal{P}_d$, interpolating between the Euclidean and the log-Euclidean metric, is the set of power Euclidean metrics that depend on a parameter $\alpha \in \mathbb{R}$ (Dryden, Pennec, et al. 2010):

$$\delta_\alpha^{\mathrm{PE}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{\alpha} \|\boldsymbol{X}^\alpha - \boldsymbol{Y}^\alpha\|_F.$$

  Since $\lim_{\alpha \to 0}(x^\alpha - 1)/\alpha = \log(x)$, one has that this family tends to the log-Euclidean metric for $\alpha = 0$ and to the Euclidean metric for $\alpha = 1$.
- **Kullback-Leibler divergence:** One can compute the Kullback-Leibler divergence (Kullback and Leibler 1951) between two centered multivariate Gaussian distributions

$\mathcal{N}(0, \boldsymbol{X})$ and $\mathcal{N}(0, \boldsymbol{Y})$. This leads to the following divergence between two SPD matrices:

$$D^{\mathrm{KL}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2}\left(\mathrm{tr}(\boldsymbol{X}^{-1}\boldsymbol{Y}) - d + \log\left(\frac{\det(\boldsymbol{X})}{\det(\boldsymbol{Y})}\right)\right).$$

The Kullback-Leibler divergence is not a distance as it is neither symmetric with respect to its two arguments nor does it satisfy the triangle inequality. It can be symmetrized to give the following Jeffreys divergence.

- **Jeffreys divergence:** This divergence is sometimes also called the J-divergence (Jeffreys 1946):

$$D^{\mathrm{J}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2}\Big(\mathrm{tr}(\boldsymbol{X}^{-1}\boldsymbol{Y}) + \mathrm{tr}(\boldsymbol{Y}^{-1}\boldsymbol{X})\Big) - d.$$

While being symmetric, this is still a divergence and not a distance.

- **S-divergence:** Another way to symmetrize the Kullback-Leibler divergence gives the S-divergence also called the Jensen-Shannon divergence or Stein divergence (J. Lin 1991; Briët and Harremoës 2009; F. Nielsen and Boltz 2011; Sra 2016b).

$$D^{\mathrm{S}}(\boldsymbol{X}, \boldsymbol{Y}) = \log\det\left(\frac{\boldsymbol{X}+\boldsymbol{Y}}{2}\right) - \frac{1}{2}\log\det(\boldsymbol{X}\boldsymbol{Y}).$$

This divergence does not satisfy the triangular inequality criterion. However, its square root has been shown to be a distance (Sra 2016b). The S-divergence between two SPD matrices corresponds to the Bhattacharyya divergence between them (Bhattacharyya 1943; Sra 2016b).

The Power Euclidean metrics come from a Riemannian structure, as they are the pullback of the Frobenius metric *via* the map $\boldsymbol{X} \mapsto \boldsymbol{X}^p$. However, the three other divergences (KL, Jeffreys, JS) come from the information geometry world (S. Amari 1997) and are related to Bregman divergences (Bregman 1967). We refer to S. Chevallier et al. (2021) for a more comprehensive review on these distances and divergences.

Finally, we have mainly focused on specific metrics defined on the manifold $\mathcal{P}_d$ of SPD matrices. In his thesis, Yann Thanwerdas characterized a wide class of Riemannian metrics that are $\mathrm{O}(d)$-invariant. This class is characterized by three continuous multivariate functions $\alpha, \beta, \gamma \colon \mathbb{R}^d \to \mathbb{R}$ satisfying three conditions of symmetry, compatibility and positivity (see Theorem 5.1 of Thanwerdas and Pennec 2023b). Thus, the class of $\mathrm{O}(d)$-invariant metrics is wider than the class of affine invariant metrics that was only characterized by two real parameters (see Remark 2.23).

2.2.4 A vectorization of the tangent spaces

As we saw in Proposition 2.12, the manifold $\mathcal{P}_d$ of SPD matrices is of dimension $d(d+1)/2$, that we denote D in the following. Thus, there exists an isomorphism between $T_{\boldsymbol{X}}\mathcal{P}_d$, the tangent space at $\boldsymbol{X} \in \mathcal{P}_d$ and $\mathbb{R}^D$. This isomorphism is not unique, and, in this section, we explain how to define such an isomorphism (we will see that we actually build an isometry) consistently for all tangent spaces of $\mathcal{P}_d$ that we call a *vectorization*. This will be useful in the following chapters to define the wrapped Gaussian distribution on $\mathcal{P}_d$. The vectorization given here depends on the metric chosen for $\mathcal{P}_d$, and we will start by defining it in a general setting[11], and then give an example for the AIRM metric defined in Definition 2.13, as it will be the main metric used in the following chapters.

In order to build the vectorization, we start by building it for $T_{I_d}\mathcal{P}_d$[12] and transport it to the other tangent spaces. We consider an orthonormal basis $\mathcal{B}^{(I_d)} = (\boldsymbol{E}_i^{I_d})_{1 \leqslant i \leqslant D}$ of $T_{I_d}\mathcal{P}_d$

[11] Note that this construction could be reproduced for any Riemannian manifold $\mathcal{M}$.

[12] It could be any other tangent space $T_{\boldsymbol{X}_0}\mathcal{P}_d$, but the one at I_d is convenient because of the simplicity of the AIRM metric at this point.

with respect to the Riemannian metric $\langle \cdot, \cdot \rangle_{I_d}$. We then first define the vectorization on $T_{I_d}\mathcal{P}_d$ and extend it to all tangent spaces of $\mathcal{P}_d$ through parallel transport.

Definition 2.31 (Vectorization of the tangent spaces)**.** We start by defining the vectorization of the tangent space $T_{I_d}\mathcal{P}_d$ at the reference point I_d by:

$$\forall \boldsymbol{U} \in T_{I_d}\mathcal{P}_d,\ \mathrm{Vect}_{I_d}(\boldsymbol{U}) = (\langle \boldsymbol{U}, \boldsymbol{E}_i^{I_d} \rangle_{I_d})_{1 \leqslant i \leqslant D} \in \mathbb{R}^D.$$

Then, for a point $\boldsymbol{X} \in \mathcal{P}_d$, we define the vectorization of the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$ by:

$$\forall \boldsymbol{U} \in T_{\boldsymbol{X}}\mathcal{P}_d,\ \mathrm{Vect}_{\boldsymbol{X}}(\boldsymbol{U}) = \mathrm{Vect}_{I_d}(\Gamma_{\boldsymbol{X} \to I_d}(\boldsymbol{U})),$$

where $\Gamma_{\boldsymbol{X} \to I_d} : T_{\boldsymbol{X}}\mathcal{P}_d \to T_{I_d}\mathcal{P}_d$ is the parallel transport from $\boldsymbol{X}$ to I_d along the (unique) geodesic connecting $\boldsymbol{X}$ and I_d.

The vectorization at I_d is simply the coefficients of the tangent vector $\boldsymbol{U}$ in the orthonormal basis $\mathcal{B}^{(I_d)}$. Let us give an important property of the vectorization:

Proposition 2.32. *The vectorization* $\mathrm{Vect}_{\boldsymbol{X}}$ *is a linear isometry between* $(T_{\boldsymbol{X}}\mathcal{P}_d, \langle \cdot, \cdot \rangle_{\boldsymbol{X}})$ *and* $(\mathbb{R}^D, \langle \cdot, \cdot \rangle_2)$.

This proposition shows that there is no volume distortion when using the vectorization.

Proof. As the parallel transport $\Gamma_{\boldsymbol{X} \to I_d}$ is a linear isometry (see Proposition 10.36 of Boumal 2023), we only need to show that Vect_{I_d} is a linear isometry. The linearity is clear from the definition. The isometry property follows from the fact that $\mathcal{B}^{(I_d)}$ is orthonormal. □

In the specific case of the AIRM geometry on $\mathcal{P}_d$, both the orthonormal basis $\mathcal{B}^{(I_d)}$ and the parallel transport admit closed-form expressions, leading to explicit formulas for the vectorization. In particular, since $T_{I_d}\mathcal{P}_d$ coincides with the Euclidean space $\mathcal{S}_d$ equipped with the Frobenius inner product, an orthonormal basis $\mathcal{B}^{(I_d)} = (\boldsymbol{E}_{ij}^{I_d})_{i \leqslant j}$ is given by:

$$\boldsymbol{E}_{ij}^{I_d} = \begin{cases} \frac{1}{\sqrt{2}}(e_{ij} + e_{ji}) & \text{for } i < j, \\ e_{ii} & \text{for } i = j \end{cases}, \tag{2.6}$$

where e_{ij} is the $d \times d$ matrix with a 1 at position (i, j) and zeros everywhere else. Then, one can compute the following expression for the vectorization.

Proposition 2.33 (Explicit vectorization on $(\mathcal{P}_d, \mathrm{AIRM})$)**.** *Denoting by* U_{ij} *the* (i, j)*-th entry of* $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d$*, the vectorization at* I_d *reads:*

$$\mathrm{Vect}_{I_d}(\boldsymbol{U}) = (U_{11}, \sqrt{2}U_{12}, U_{22}, \sqrt{2}U_{13}, \sqrt{2}U_{23}, U_{33}, \ldots, \sqrt{2}U_{d-1,d}, U_{dd}) \in \mathbb{R}^D.$$

Moreover, using the parallel transport formula $\Gamma_{\boldsymbol{X} \to I_d}(\boldsymbol{U}) = \boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2}$*, the vectorization at a general point* $\boldsymbol{X} \in \mathcal{P}_d$ *reads:*

$$\mathrm{Vect}_{\boldsymbol{X}}(\boldsymbol{U}) = \mathrm{Vect}_{I_d}(\boldsymbol{X}^{-1/2}\boldsymbol{U}\boldsymbol{X}^{-1/2}).$$

Proof. The formula for Vect_{I_d} follows from computing the inner products with the basis elements $\mathcal{B}^{(I_d)} = (\boldsymbol{E}_{ij}^{I_d})_{i \leqslant j}$. At I_d, the AIRM coincides with the Frobenius inner product, so $\langle \boldsymbol{U}, \boldsymbol{E}_{ij}^{I_d} \rangle_{I_d} = \mathrm{tr}(\boldsymbol{U}\boldsymbol{E}_{ij}^{I_d})$. For $i < j$, one has $\boldsymbol{E}_{ij}^{I_d} = \frac{1}{\sqrt{2}}(e_{ij} + e_{ji})$, so $\mathrm{tr}(\boldsymbol{U}\boldsymbol{E}_{ij}^{I_d}) = \frac{1}{\sqrt{2}}(U_{ji} + U_{ij}) = \sqrt{2}U_{ij}$. For $i = j$, $\boldsymbol{E}_{ii}^{I_d} = e_{ii}$, so $\mathrm{tr}(\boldsymbol{U}\boldsymbol{E}_{ii}^{I_d}) = U_{ii}$. The formula for $\mathrm{Vect}_{\boldsymbol{X}}$ then follows from setting $\boldsymbol{Y} = I_d$ in the parallel transport formula of Proposition 2.20. □

This vectorization is also used a lot to help computations as it provides a minimal representation in an orthonormal basis (Tuzel et al. 2008; Barachant, Bonnet, et al. 2011; Yger, Berar, et al. 2017) and is also described in Pennec (2020).

2.3 Related work

In this section, our goal is to give a vast panorama of the different tools that are used to do learning on a manifold. The different tools presented have been sorted into different categories: supervised learning, kernel methods, dimension reduction tools, probability distributions on manifolds and deep methods. A comprehensive and recent survey on ML methods with geometric, topological, and algebraic structures is proposed in Papillon et al. (2025).

Let us start by making a clear distinction between two different, yet closely related, frameworks: *manifold learning* and *learning on a manifold*. The first one, *manifold learning* is, as stated in Meilă and H. Zhang (2024): "Manifold learning, also known as nonlinear dimension reduction, is a set of methods to find the low-dimensional structure of data." Thus, the goal of manifold learning is to "reveal the geometric shape of high-dimensional point clouds and allow one to visualize, denoise, and interpret them." When the data of interest is too high-dimensional, one needs to first reduce its dimension in order to learn useful features from it. The toolbox of manifold learning contains tools such as *Principal Component Analysis* (PCA) (Jolliffe 2002), *Locally Linear Embedding* (LLE) (S. T. Roweis and Saul 2000), *Isomap* (Tenenbaum et al. 2000), *t-distributed Stochastic Neighbor Embedding* (t-SNE) (Maaten and Hinton 2008) or *UMAP* (McInnes et al. 2018) to only cite a few of the most popular. While t-SNE will be of interest in Chapter 7, where we study a specific version of the t-SNE algorithm for SPD matrices, the methods we focus on in this manuscript do not fall within the scope of manifold learning. Rather, throughout the rest of this manuscript, the underlying manifold will always be known, and our goal will be to develop and study learning tools and methods that operate on manifold-valued data while taking their geometric structure into account. This is what we call *learning on a manifold*. The main manifold we will focus on is the manifold of *Symmetric Positive Definite matrices* (SPD), denoted $\mathcal{P}_d$, that has been thoroughly introduced in Section 2.2, but we will also encounter throughout this manuscript the hyperbolic space denoted $\mathbb{H}^d$ or a general manifold denoted $\mathcal{M}$.

Let us now focus on *learning on a manifold* and give a panorama of the existing tools used to learn features out of manifold-valued points.

2.3.1 Supervised learning

The problem of supervised learning is well described by Bach (2024): "Given some observations $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ [...] the main goal of supervised learning is to predict a new $y \in \mathcal{Y}$ given a new previously unseen $x \in \mathcal{X}$." In our setting of *learning on a manifold*, the input space $\mathcal{X}$ will most often be a Riemannian manifold, and in the context of Motor Imagery Brain-Computer interfaces (BCI), the labels y will often be the mental task that the subject is asked to perform (`right hand`, `left hand`, `feet`, `tongue`...).

In BCI, the Riemannian geometry of SPD matrices has been introduced by Barachant (2012) for classification purposes. The first Riemannian algorithm proposed by Barachant is the *Minimum Distance to the Mean* (MDM) algorithm in Barachant, Bonnet, et al. (2010b). The principle behind this algorithm is very simple: during the training phase, the Riemannian mean of each class is computed. A new unseen covariance matrix is then classified based on its minimal Riemannian distance to the class means. This algorithm is an instance of a *prototype method* where the prototype (see chapter 13 of Hastie et al. 2009) is the Riemannian

mean of each class. In the same article, the authors also propose a geodesic filtering process achieved in the tangent space using a Linear Discriminant Analysis (LDA). The matrices are then projected back to the manifold and classified with a regular MDM. This pipeline is called *FgMDM* and achieves better classification accuracy than regular MDM (see Figure 6 in Barachant, Bonnet, et al. (2010b)). Then, in Barachant, Bonnet, et al. (2011), the authors developed *Tangent Space methods*, in which they project the points from the manifold into a well-chosen tangent space (often the tangent space at the Riemannian mean), and apply classical learning algorithms such as Linear Discriminant Analysis (LDA, see Section 4.3 of Hastie et al. 2009) or Support Vector Machines (SVM, see Section 12.3 of Hastie et al. 2009). Regression models have also been extended to the manifold of SPD matrices (Sabbagh et al. 2019; Mellot, Collas, S. Chevallier, Gramfort, et al. 2024). They mostly rely on a tangent space projection, and they apply a Euclidean regression model in the tangent space. A fully Riemannian regression method has also been proposed by Thomas Fletcher (2011), who relies on geodesics to perform geodesic regression. Let $y_1, ..., y_N \in \mathcal{M}$ where $\mathcal{M}$ is a Riemannian manifold and $x_1, ..., x_N \in \mathbb{R}$ their associated scalar values. The goal of geodesic regression is to find a point $\hat{p} \in \mathcal{M}$ and a tangent vector $\hat{v} \in T_p\mathcal{M}$ such that the geodesic $\gamma(t) = \text{Exp}_p(tv)$ on $\mathcal{M}$ best models the relationship between the x_i and the y_i. This can be modeled by:

$$(\hat{p}, \hat{v}) = \underset{(p,v)}{\arg\min} \frac{1}{2} \sum_{i=1}^{N} \delta(\text{Exp}_p(x_i v), y)^2. \tag{2.7}$$

However, estimating this model can be computationally heavy in practice, as to solve the optimization problem given in Equation (2.7), one needs to compute the derivative of the Riemannian exponential mapping, which relies on Jacobi fields (see Chapter 10 of J. M. Lee 2018). Polynomial regression on manifolds (Hinkle et al. 2012) as well as Bezier-splines fitting on manifolds (Hanik et al. 2020) have also been developed to bridge the gap between Euclidean and manifolds methods.

More complex classifiers have been extended to manifolds. The *Multinomial Logistic Regression* (MLR, see Section 18.3.2 of Hastie et al. 2009) has first been extended to specific manifolds, such as the manifold of SPD matrices $\mathcal{P}_d$ in Z. Chen, Yue Song, G. Liu, et al. (2024), in which the authors only consider flat metrics for $\mathcal{P}_d$, such as the Log-Euclidean metric or the Log-Cholesky one. Many authors rely on *gyrovectors* to build MLR such as X. S. Nguyen and S. Yang (2023) and X. S. Nguyen, S. Yang, and Histace (2024) for $\mathcal{P}_d$ or Ganea et al. (2018) for the hyperbolic space. The goal of using gyrovectors is to recover a structured, vector-like algebraic calculus (addition, inner products, distances) on curved non-Euclidean spaces, enabling the direct generalization of Euclidean operations to manifolds in closed form. Riemannian MLRs were mainly used as the last layer of a deep learning architecture, and we will cover in more depth the attempts to extend deep learning to a manifold in Section 2.3.5.

Some of these classifiers will be encompassed in a probabilistic framework for learning tools on the manifold $\mathcal{P}_d$ in Chapter 4. Moreover, we will study the interpretability of Riemannian classifiers in Chapter 7.

2.3.2 Kernel methods

When the data of interest lies on a Riemannian manifold, it is intrinsically non-linear. Popular methods used to solve non-linear problems are *kernel methods* (Hofmann et al. 2008; Gärtner et al. 2002; Yger 2013), that implicitly map data into a high-dimensional (often infinite-dimensional) feature space via a kernel function, which computes inner products between mapped points without ever explicitly constructing the mapping. This way, linear algorithms (like SVMs, ridge regression, or PCA) can capture nonlinear relationships in the original data. One of the main challenges when working with kernel methods is to build a well-suited kernel. A straightforward and well-studied way is to build a radial basis function kernel which only

relies on the geodesic (Riemannian) distance δ between observations:

$$k(x, y) = \exp(-\lambda \delta^q(x, y)), \tag{2.8}$$

with $\lambda, q > 0$ and x, y two observations. When $q = 2$, this kernel is called the *Gaussian kernel* and $q = 1$ gives the *Laplacian kernel*. A key property of a kernel is its *positive definiteness*. When studying such kernels on manifolds, a key result is given by Feragen et al.:

Theorem 2.34 (Feragen et al. 2015)**.** *Let $\mathcal{M}$ be a complete, smooth Riemannian manifold with its associated geodesic (Riemannian) distance δ. Then,*

- *The geodesic distance δ in a geodesic metric space (X, δ) is* conditionally negative definite[a] *if and only if the corresponding geodesic Laplacian kernel $k(x, y) = \exp(-\lambda \delta(x, y))$ is positive definite for all $\lambda > 0$.*
- *If the Gaussian kernel $k(x, y) = \exp(-\lambda \delta^2(x, y))$ is positive definite for all $\lambda > 0$, then, the Riemannian manifold $\mathcal{M}$ is isometric to a Euclidean space.*
- *Let $q > 2$. Then there is some $\lambda > 0$ so that the kernel given in Equation (2.8) is not positive definite.*

[a] Please refer to Definition 3 of Feragen et al. (2015) for the definition of a conditionally negative definite kernel.

This result illustrates the difficulty of using kernels on non-flat Riemannian manifolds, such as the manifold of SPD matrices equipped with the *Affine Invariant Riemannian Metric* (see Section 2.2 for more details on that geometry), or the hyperbolic space. To circumvent this problem, several strategies have been studied to build kernels on curved spaces. On the manifold of SPD matrices, Barachant, Bonnet, et al. (2013) rely on a tangent-space approximation to perform a *Support Vector Machine* (SVM) approach. This means that they choose a "good" tangent space (often the one at the Riemannian mean of the training dataset), project the points into this Euclidean space, and apply the SVM in this flat tangent space. In Jayasumana et al. (2013), the authors rely on flat metrics for the space of SPD matrices, such as the *Log-Euclidean metric* (Arsigny et al. 2006a) to perform some classical kernel tasks (kernel SVM, kernel PCA or kernel k-means).

Remark 2.35. When the chosen tangent space is the one at identity I_d, using a kernel in the tangent space $T_{I_d}\mathcal{P}_d$ or using the Log-Euclidean metric is equivalent. See Remark 2.28 for more details on this.

Among the other geometries for the manifold $\mathcal{P}_d$ of SPD matrices presented in Section 2.2.3.6, the Gaussian kernel using the S-divergence has been studied by Sra (2016b). According to his Theorem 3.10, the resulting function is a valid kernel if, and only if,

$$\lambda \in \left\{ \frac{j}{2} \mid j \in \mathbb{N} \text{ and } 1 \leqslant j \leqslant d-1 \right\} \bigcup \left\{ \gamma \mid \gamma \in \mathbb{R} \text{ and } \gamma \geqslant \frac{d-1}{2} \right\}.$$

More recently, Gabriel Wagner vom Berg studied a large variety of kernels with different metrics on the manifold $\mathcal{P}_d$ of SPD matrices in his PhD thesis (Wagner vom Berg 2025; Wagner vom Berg et al. 2025). In order to conduct his study, he carefully selected the parameter λ to ensure that the considered kernels are positive definite. These kernels were used to perform a classification task on a BCI dataset using an SVM and a dimension reduction task using a kernel PCA.

2.3.3 Dimension reduction

Before classifying the data, pre-processing the data is often required. The data collected can be noisy and is often very high-dimensional. A classical pre-processing step used when this happens is to reduce the dimension of the data. This is done using *dimension reduction*

algorithms. The goal of such an algorithm is to compute a low-dimensional embedding of a high-dimensional point cloud which still captures some of the similarities of the high-dimensional point cloud or some meaningful properties of the original data.

In this section, we will not focus on classical Euclidean dimension reduction algorithms, but rather their Riemannian counterparts. One of the most famous Euclidean dimension reduction algorithms is the *Principal Component Analysis* (PCA, Hotelling 1933) and this algorithm has been extended to Riemannian manifolds in different ways. One of the main adaptations of the PCA to manifolds is Fletcher's *Principal Geodesic Analysis* (PGA, P. Fletcher et al. 2004) that replaces affine subspaces with geodesic subspaces spanned by a tangent vector restricted to belong to a linear subspace of the tangent space. In practice, the procedure to compute the right geodesic subspaces is too computationally expensive (see Sommer, Lauze, and M. Nielsen 2014, for a complete implementation), so it is approximated by doing a PCA in the tangent space at the Riemannian mean of the dataset (P. T. Fletcher and Joshi 2007; Sommer, Lauze, Hauberg, et al. 2010). The points are mapped to the tangent space using the Riemannian logarithm, and the classical PCA is performed on the covariance matrix in the tangent space. This simple and efficient procedure can be enough when the data is well centered around a central value, but is not enough for multimodal or large-support distributions. In Pennec (2015b), Xavier Pennec proposes to replace the geodesic subspaces of PGA by *barycentric subspaces*, built as the set of points that can be reached by computing weighted Fréchet means[13]. A barycentric subspace is the curved-space analog of an affine subspace. Pennec then computed those barycentric subspaces and proposed to use them to perform a Riemannian PCA on the sphere in Pennec (2015a). A probabilistic version of the PGA has been proposed in M. Zhang and Tom Fletcher (2013), where the authors rely on a latent variable model in which the parameters are estimated using a Monte Carlo Expectation Maximization algorithm.

Other attempts at building a PCA for specific manifolds have been proposed, such as a *Geometry Aware PCA* for SPD matrices in Horev et al. (2016) where the authors learn an element from the Grassmann manifolds[14] (Bendokat et al. 2020) to reduce the dimension of the data while maximizing the Fréchet variance. For the hyperbolic space, the authors of Chami, Gu, et al. (2021) propose a PCA based on the projection of the data onto horospheres (level sets of Busemann functions, see Busemann 1955) while minimizing the unexplained variance of the projected data. Busemann functions provide a natural generalization of affine functions on geodesic metric spaces of non-positive curvature, and their level sets thus generalize the notion of affine hyperplanes.

PCA is not the only dimension reduction method that has been generalized to Riemannian manifolds. In A. Goh and Vidal (2008), the Locally Linear Embedding (LLE, Saul and S. T. Roweis 2003), the Laplacian Eigenmaps (Belkin and Niyogi 2001) and the Hessian LLE (Donoho and Grimes 2003) have been extended to general Riemannian manifolds. The authors implemented the different methods for the manifold of SPD matrices and tested them on DTI data.

Another important dimension reduction algorithm is the *t-distributed stochastic neighbor embedding*, or *t-SNE*, developed by Laurens van der Maaten and Geoffrey Hinton in Maaten and Hinton (2008). This algorithm will be at the heart of Chapter 7, but let us already give some of its Riemannian extensions. In Bergsson and Hauberg (2022), the authors extend the original *stochastic neighbor embedding* (SNE, Hinton and S. Roweis 2002) on any Riemannian manifold. For this, they use the density of a Brownian motion on the Riemannian manifold (the heat kernel approximation) to compute the similarities in the high and the low-dimensional spaces. Another original aspect of their work is that the low-dimensional embedding space is

[13]The barycentric subspace spanned by $p_0, \dots, p_k$ is defined as: $\mathrm{BS}(p_0, \dots, p_k) = \{\text{Weighted Fréchet means of } p_0, \dots, p_k \text{ with weights } \lambda_i,\ \sum_i \lambda_i = 1\}$.

[14]The set of all linear subspaces of fixed dimension p of the Euclidean space $\mathbb{R}^d$ forms the Grassmann manifold $\mathrm{Gr}(d, p)$.

flexible, and one can choose it depending on the geometry of the high-dimensional data ($\mathbb{R}^2$, the sphere or any other manifold). Most of the other extensions of t-SNE focus on a specific manifold. For the hyperbolic manifold, several authors proposed an extension such as Y. Guo et al. (2022), Klimovskaia et al. (2020) or Y. Zhou and Sharpee (2021). A specific study has been conducted in Skrodzki et al. (2024) to accelerate the hyperbolic t-SNE, as is frequently done for the classical t-SNE algorithm. For the manifold of SPD matrices, a tangent space version of the t-SNE has been studied in J. Liu et al. (2026).

One cannot talk about modern dimension reduction algorithms without mentioning *UMAP* proposed in McInnes et al. (2018) to visualize high-dimensional data. This algorithm is meant to be used for data lying on a low-dimensional, unknown manifold and tries to model it using algebraic topology. Then, it optimizes a low-dimensional graph to match the high-dimensional one. In practice this pushes connected points together and repels disconnected ones, similar in spirit to t-SNE but using a different kernel and loss.

Other specific tools have been developed to reduce the dimensionality of a point cloud that lives on a known Riemannian manifold. For instance, for SPD matrices, Harandi et al. (2014) proposed a method similar to that of Horev et al. (2016), adding an affinity matrix that exploits class information to minimize the intra-class distances while simultaneously maximizing the inter-class distances. BCI is an interesting motivation to build dimension reduction algorithms for SPD matrices as the dimension of the covariance matrices that are dealt with grows quadratically with the number of electrodes used to record the EEG. A lot of algorithms have been built with this application in mind (Coelho Rodrigues et al. 2017; Davoudi et al. 2017; Tanaka et al. 2016; Z. Peng et al. 2023; Xie et al. 2017).

All of those dimension reduction algorithms can be divided into two types: algorithms made for visualization purposes and algorithms made for feature extraction. In the first category of algorithms, one can find the t-SNE and its Riemannian counterparts or UMAP. The goal of such algorithms is to give qualitative insights into the high-dimensional dataset, and they are often used to better understand the data one is dealing with. The second type of algorithms, in which one can find the Riemannian PCA of Horev et al. (2016) or Harandi et al. (2014), but also most of the BCI-specific dimension reduction algorithms, is used as a first tool in a classification pipeline, where the reduction of the dimension helps with reducing the noise and the computational complexity.

2.3.4 Probability distributions on manifolds

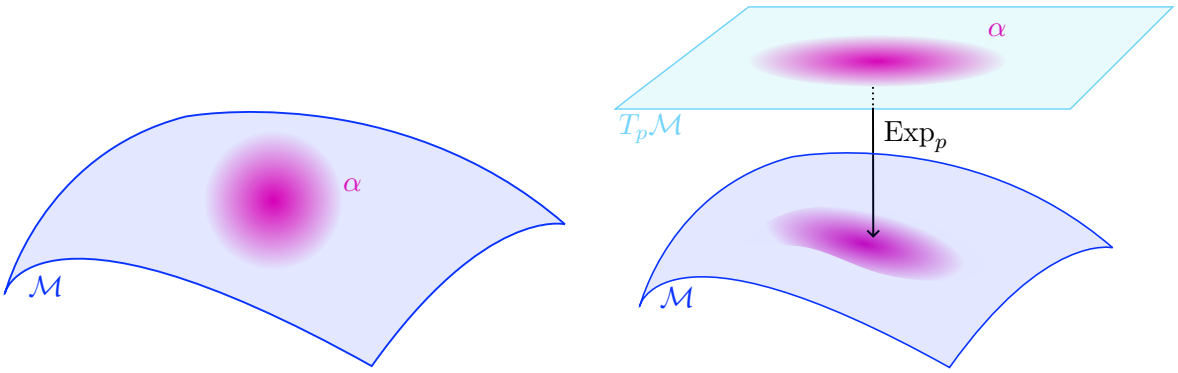
The definition and study of probability distributions on the Riemannian manifold of SPD matrices constitute the main focus of this manuscript, we therefore provide here a review of the existing literature on this topic. We do not focus only on the manifold $\mathcal{P}_d$ of SPD matrices, as this topic has been studied for larger classes of manifolds.

2.3.4.1 Gaussian distributions on general manifolds

Several works have already considered probability distributions on a Riemannian manifold. When restricting ourselves to the Gaussian distribution, two main works stand out: the isotropic Gaussian defined by Said et al. on Riemannian symmetric spaces introduced in Said, Hajri, et al. (2018) and the entropy-maximizing Gaussian of Pennec introduced in Pennec (2006). The first Gaussian is isotropic and relies on two parameters: a center of mass $\bar{\boldsymbol{X}}$ and a dispersion parameter $\sigma > 0$:

$$f_{\bar{\boldsymbol{X}},\sigma}(\boldsymbol{X}) = \frac{1}{\zeta(\sigma)} \exp\left(-\frac{\delta(\boldsymbol{X}, \bar{\boldsymbol{X}})^2}{2\sigma^2}\right), \tag{2.9}$$

where $\zeta(\sigma) = \int_{\mathcal{M}} \exp\left(-\delta(\boldsymbol{X}, \bar{\boldsymbol{X}})^2/2\sigma^2\right) \mathrm{dvol}(\boldsymbol{X})$ is the normalizing constant. This distribution is illustrated in Figure 2.9a. In their paper, they study the density of the isotropic



(a) Isotropic distribution on $\mathcal{M}$.

(b) Wrapped distribution on $\mathcal{M}$.

Figure 2.9. Illustration of two approaches for defining probability distributions on a manifold $\mathcal{M}$. The one on the left is an isotropic distribution α defined intrinsically on the manifold $\mathcal{M}$, as it is done in Said, Hajri, et al. (2018) or in R. A. Fisher (1953). The figure on the right is an illustration of a wrapped distribution $\mathrm{Exp}_p \# \alpha$, where α is defined on the tangent space at $p \in \mathcal{M}$ and then pushed forward to the manifold using Exp_p.

Gaussian, give a sampling procedure (which requires sampling from a non-trivial multivariate density) and give the Maximum Likelihood Estimators (MLE) of the two parameters. While the estimator $\hat{\boldsymbol{X}}_N$ of the center of mass $\bar{\boldsymbol{X}}$ relies on the Fréchet mean, which can be computed using a gradient descent algorithm (see Section 4.6 of Pennec 2006), the MLE of the dispersion parameter σ involves solving the following equation:

$$\sigma^3 \frac{\mathrm{d}}{\mathrm{d}\sigma} \log \zeta(\sigma) = \frac{1}{N} \sum_{i=1}^{N} \delta(\hat{\boldsymbol{X}}_N, \boldsymbol{X}_i)^2. \tag{2.10}$$

This highly non-linear scalar equation is not trivial to solve analytically, but Newton's method can for example be used to solve it numerically. However, this non-linear equation involves the normalizing constant of the distribution and computing it requires performing a Monte-Carlo approximation. They study this specific distribution on the manifold of SPD matrices in Said, Bombrun, et al. (2017) and for Toeplitz covariance matrices and Block-Toeplitz covariance matrices in Said, Hajri, et al. (2018).

Pennec's Gaussian is built using the characterization of the Gaussian distribution as the distribution that maximizes entropy given a mean $\bar{\boldsymbol{X}}$ and a concentration matrix Γ (Theorem 13.2.2 of Kagan et al. 1973). The resulting density is

$$f_{\bar{\boldsymbol{X}},\Gamma}(\boldsymbol{X}) = k \exp\left(-\frac{1}{2} \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})^\top \Gamma \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})\right), \tag{2.11}$$

where k is the normalizing constant $k^{-1} = \int_{\mathcal{M}} \exp\left(-\operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})^\top \Gamma \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})/2\right) \mathrm{dvol}(\boldsymbol{X})$. The concentration matrix Γ is related to the covariance Σ by:

$$\Sigma = k \int_{\mathcal{M}} \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})^\top \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X}) \exp\left(-\frac{1}{2} \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})^\top \Gamma \operatorname{Log}_{\bar{\boldsymbol{X}}}(\boldsymbol{X})\right) \mathrm{dvol}(\boldsymbol{X}).$$

As stated in Pennec (2006), "From the concentration matrix, we can compute the covariance of the random point, at least numerically, but the reverse is more difficult." This anisotropic Gaussian can thus be hard to manipulate in practice: there is no closed-form expression for the density (it relies on an integral over the whole manifold), no sampling procedure is known to us, and the estimation of the parameters in the case of a full covariance matrix has not been studied.

This entropy-maximizing Gaussian is a generalization of the isotropic Gaussian defined in Equation (2.9) to the anisotropic case. Indeed, if $\Sigma = \sigma^2 I_d$, with $\sigma > 0$, then the entropy-maximizing Gaussian given in Equation (2.11) reduces to the isotropic one.

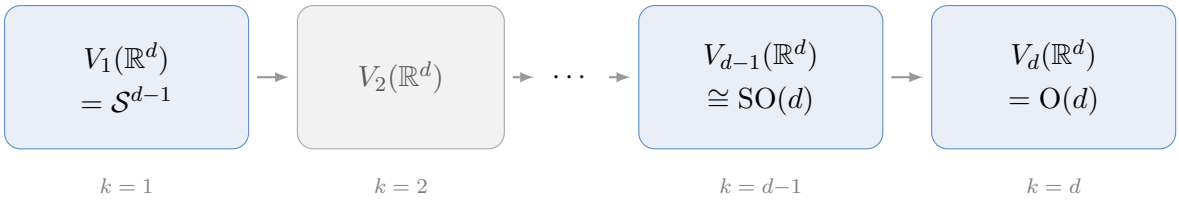


Figure 2.10. The Stiefel manifold $V_k(\mathbb{R}^d)$ as a common generalization of the sphere ($k = 1$), the special orthogonal group ($k = d - 1$), and the orthogonal group ($k = d$).

2.3.4.2 The von Mises-Fisher distribution and its generalizations

A manifold that has attracted interest for the study of probability distributions is the sphere $\mathcal{S}^{d-1}$. This manifold has led to an entire field of statistics: *directional statistics*. An important reference for this field is the book by Mardia and Jupp (2009) that studies the main distributions defined on $\mathcal{S}^{d-1}$. The von Mises-Fisher is the dominant parametric model on the hypersphere (R. A. Fisher 1953) and resembles a Gaussian. It is parametrized by a mean direction $\boldsymbol{\mu} \in \mathcal{S}^{d-1}$ and a concentration parameter $\kappa \geqslant 0$, that characterizes how strongly unit vectors are concentrated at the mean direction:

$$\forall\, \boldsymbol{x} \in \mathcal{S}^{d-1},\ f_{\boldsymbol{\mu},\kappa}(\boldsymbol{x}) = C_p(\kappa) \exp(\kappa \boldsymbol{\mu}^\top \boldsymbol{x}),$$

where $C_p(\kappa)$ is the normalizing constant and is equal to

$$C_p(\kappa) = \frac{\kappa^{p/2-1}}{(2\pi)^{p/2} I_{p/2-1}(\kappa)},$$

where I_v denotes the modified Bessel function of the first kind at order v (Robert 1990). This distribution is isotropic. The von Mises–Fisher distribution is uniquely characterized as the maximum entropy distribution on the unit sphere, subject to a fixed constraint on the first moment (Rao 2009, pp. 172-174). If anisotropy is required, one can use the Fisher–Bingham family (Mardia 1972), the canonical exponential family on the sphere or the Kent distribution (Kent 1982), a simpler version of the Fisher–Bingham distribution, that still preserves anisotropy. The density of the Fisher–Bingham distribution depends on three parameters: $\boldsymbol{\mu}$ and κ, that play a similar role as for the von Mises-Fisher distribution, but also a $(d+1) \times (d+1)$ symmetric matrix $\boldsymbol{A}$ that will control the shape of the probability contours on the sphere. The expression of the density is then:

$$\forall\, \boldsymbol{x} \in \mathcal{S}^{d-1},\ f_{\boldsymbol{\mu},\kappa,\boldsymbol{A}} \propto \exp(\kappa \boldsymbol{\mu}^\top \boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x}).$$

Imposing $\boldsymbol{A}\boldsymbol{\mu} = 0$ in the previous density will lead to the Kent distribution. Hauberg has also proposed an anisotropic version of the Gaussian on the sphere in Hauberg (2018). The von Mises-Fisher distribution has been used to perform clustering on high-dimensional text and gene-expression data in Banerjee et al. (2005). For this, the authors rely on a mixture of von Mises–Fisher distributions on the sphere $\mathcal{S}^{d-1}$. A more thorough review on probability distributions defined on the sphere and their applications in machine learning can be found in Sra (2016a) or in Pewsey and García-Portugués (2021).

Recall that the Stiefel manifold $V_k(\mathbb{R}^d)$ is the set of all orthonormal k-frames in $\mathbb{R}^d$:

$$V_k(\mathbb{R}^d) = \{\boldsymbol{X} \in \mathbb{R}^{d \times k} \mid \boldsymbol{X}^\top \boldsymbol{X} = I_k\} \quad (0 < k \leqslant d).$$

This manifold generalizes the sphere $\mathcal{S}^{d-1}$, which corresponds to the particular case of a single unit vector. In other words, one has $V_1(\mathbb{R}^d) = \mathcal{S}^{d-1}$. Conversely, when $k = d$, the Stiefel manifold coincides with the orthogonal group: $V_d(\mathbb{R}^d) = \mathrm{O}(d)$. Moreover, one can

identify $\mathrm{SO}(d)$, the special orthogonal group, i.e. the set of $d \times d$ orthogonal matrices of determinant 1, with $V_{d-1}(\mathbb{R}^d)$: for $\boldsymbol{X} \in \mathrm{SO}(d)$, the last column is uniquely determined by the other $d-1$ columns, which form an orthonormal $(d-1)$-frame. These relationships between the sphere $\mathcal{S}^{d-1}$, the Stiefel manifold, $\mathrm{O}(d)$ and $\mathrm{SO}(d)$ are illustrated in Figure 2.10. One can thus generalize the von Mises-Fisher distribution, defined on the sphere S^{d-1}, to the Stiefel manifold $V_k(\mathbb{R}^d)$ (Downs 1972) and then to $\mathrm{O}(d)$ and $\mathrm{SO}(d)$ (Sei et al. 2013). On those spaces, the von Mises-Fisher distribution is also called the *matrix Langevin distribution* (Chikuse 2012; Pal et al. 2020). On the Stiefel manifold $V_k(\mathbb{R}^d)$, the density of the matrix Langevin distribution is

$$\forall\, \boldsymbol{X} \in V_k(\mathbb{R}^d),\ \ f_{\boldsymbol{F}}(\boldsymbol{X}) \propto \exp(\mathrm{tr}(\boldsymbol{F}^\top \boldsymbol{X})),$$

where $\boldsymbol{F}$ is a $d \times k$ matrix. This is the direct matrix analog of the von Mises-Fisher density $\exp(\kappa \boldsymbol{\mu}^\top \boldsymbol{x})$: the inner product $\kappa \boldsymbol{\mu}^\top \boldsymbol{x}$ is replaced by the Frobenius inner product $\mathrm{tr}(\boldsymbol{F}^\top \boldsymbol{X})$. Just as the Fisher–Bingham family adds a quadratic term to the von Mises-Fisher on the sphere, one can add a quadratic term $\mathrm{tr}(\boldsymbol{X}^\top \boldsymbol{B} \boldsymbol{X})$ to the matrix Langevin on the Stiefel manifold, yielding the *matrix Bingham* or *matrix Fisher–Bingham distribution*.

Then, those distributions can be generalized to the Grassmann manifold $\mathrm{Gr}(d, k)$, the set of all linear subspaces of fixed dimension k of $\mathbb{R}^d$ (Bendokat et al. 2020), as it can be thought of as the quotient space $V_k(\mathbb{R}^d)/O(k)$. For more details on distributions on the Stiefel or Grassmann manifold, we refer to chapter 13 of Mardia and Jupp (2009) or on the book on this specific topic Chikuse (2012).

On the hyperbolic space $\mathbb{H}^d$, the hyperboloid distribution was constructed by analogy with the von Mises-Fisher (vMF) distribution on the sphere. It was introduced in Barndorff-Nielsen (1977). Its density with respect to the Riemannian volume measure on $\mathbb{H}^d$ is proportional to $\exp(-\kappa \langle \mu, x \rangle_{\mathcal{L}})$, where $\mu \in \mathbb{H}^d$ and $\kappa > 0$. Other distributions have been studied on the hyperbolic space, such as the Souriau-Gibbs distributions introduced in Barbaresco (2019) or the Poincaré distribution in F. Nielsen and Okamura (2024).

2.3.4.3 Probability distributions on $\mathcal{P}_d$

On the manifold of SPD matrices $\mathcal{P}_d$, one can distinguish between two types of distributions: the distributions that do not take into account the Riemannian geometry of $\mathcal{P}_d$ and the distributions that do. For the first category, one of the most classical probability distributions on the set of SPD matrices is the *Wishart distribution* $\mathcal{W}_d(k, \Sigma)$, where d denotes the matrix dimension and k represents the degrees of freedom (Wishart 1928). This distribution characterizes the law of a random $d \times d$ matrix $\boldsymbol{X} = \boldsymbol{M}\boldsymbol{M}^\top$, where $\boldsymbol{M}$ is a $d \times k$ matrix whose k columns are independently drawn from a d-variate normal distribution with zero mean, $\mathcal{N}(0_d, \Sigma)$. Its density is

$$\forall\, \boldsymbol{X} \in \mathcal{P}_d,\ \ f_{k,\Sigma}(\boldsymbol{X}) = \frac{1}{2^{dk/2} \det \Sigma^{d/2} \Gamma_k\left(\frac{d}{2}\right)} \det(\boldsymbol{X})^{(d-k-1)/2} e^{-\frac{1}{2}\mathrm{tr}(\Sigma^{-1}\boldsymbol{X})},$$

where Γ_k is the multivariate gamma function (Muirhead 1982). The Wishart distribution serves as a multivariate generalization of the gamma distribution. Notably, in the univariate case where $d = 1$ and $\Sigma = 1$, this distribution reduces to a chi-squared distribution $\chi^2(k)$ with k degrees of freedom. The Wishart distribution is often used in Bayesian statistics, when estimating a multivariate Gaussian distribution, as the conjugate prior to the precision matrix Σ^{-1}, where Σ is the covariance matrix. This distribution is technically defined on the Euclidean space of symmetric matrices and its support is the SPD cone. Thus, it does not account for the Riemannian geometry of $\mathcal{P}_d$. The Wishart distribution has been extended to the t-Wishart distribution in Ayadi et al. (2023) and to an elliptical version in Ayadi et al. (2024), leading to a more robust and flexible distribution.

For distributions defined on $\mathcal{P}_d$ that take into account its Riemannian geometry, the most popular and most used is the isotropic Gaussian given in Equation (2.9), studied specifically on $\mathcal{P}_d$ in Said, Bombrun, et al. (2017). A Laplace version of this distribution (with a density $\propto \exp(-\delta(\boldsymbol{X}, \bar{\boldsymbol{X}})/\sigma^2)$) has also been introduced in Hajri et al. (2016) to provide robustness. Finally, mixtures of Gaussians have been studied on the manifold of SPD matrices, where each component is an isotropic Gaussian in Said, Bombrun, et al. (2017) or another type of Gaussian in Jaquier and Calinon (2017) that resembles the one proposed in Pennec (2006).

2.3.4.4 Wrapped distributions

In the previous paragraph, we described distributions that were intrinsically defined on the manifold. They were most of the time inspired by the Gaussian density, or at least by the definition of an exponential family, and the density was adapted to work on each manifold. However, there is another way of defining a probability distribution on a Riemannian manifold $\mathcal{M}$, and that relies on the Euclidean nature of the tangent space $T_p\mathcal{M}$ where $p \in \mathcal{M}$. One can thus consider a classical Euclidean distribution α on the tangent space $T_p\mathcal{M}$ and, using the Riemannian exponential mapping $\mathrm{Exp}_p \colon T_p\mathcal{M} \to \mathcal{M}$, one can *wrap* it onto the manifold $\mathcal{M}$:

$$\mathrm{Exp}_p \# \alpha,$$

where $\#$ denotes the push-forward (see Section 3.6 of Bogachev 2007). This is called a *wrapped distribution* and a *wrapped Gaussian* defined on the manifold $\mathcal{P}_d$ of SPD matrices will be at the core of Chapter 3. An illustration of a wrapped distribution is given in Figure 2.9b. Let us now describe different works that also study wrapped distributions.

There are different ways of wrapping a distribution from a tangent space onto the manifold. In our work, we focus on exponentially-wrapped densities, meaning we will use the Riemannian exponential map to push densities supported on a tangent space to the manifold. Such exponentially-wrapped densities have been studied on specific manifolds, where the resulting distribution enjoys interesting properties. On the circle S^1, as the Riemannian exponential is simply $\mathrm{Exp}_0(\boldsymbol{x}) = \boldsymbol{x} \pmod{2\pi}$, if we denote by f_E the Euclidean density of the distribution on the tangent space at 0, the density $f_{\mathcal{S}^1}$ of the wrapped distribution on the circle $\mathcal{S}^1$ will be (see section 3.5.7 of Mardia and Jupp 2009; Collett and Lewis 1981):

$$f_{\mathcal{S}^1}(\boldsymbol{x}) = \sum_{k=-\infty}^{\infty} f_E(\boldsymbol{x} + 2k\pi). \tag{2.12}$$

Then, one can consider different distributions to wrap. In Mardia and Jupp (2009), the authors study the wrapped Poisson, the wrapped Normal and the wrapped Cauchy distributions. The wrapped Cauchy distribution on the sphere has also been studied in more depth in Kato and McCullagh 2020. On the hyperbolic space, the wrapped Gaussian has been studied, first by Nagano et al. (2019) and then in Mathieu et al. (2019) and in Cho et al. (2022). Their primary use of this wrapped Gaussian is to model the latent space of a Variational Autoencoder, which in turn is used to learn the distribution's parameters. None of the mentioned work on hyperbolic spaces focuses on theoretical properties of the wrapped Gaussian. Wrapped Gaussians have also been studied on Lie groups in Falorsi et al. (2019). An important observation is that, in all the previously mentioned works, the Gaussians that are being wrapped are constrained to be centered in the tangent space, meaning that their mean in the tangent space is fixed at 0:

$$\mathrm{Exp}_p \# \mathcal{N}(0, \Sigma).$$

In contrast, in Chapter 3, we adopt a more general framework by allowing the wrapped Gaussian distribution on the manifold of SPD matrices to have a nonzero mean in the tangent space.

Distribution	Manifold	Anisotropy	Closed-form
Isotropic Gaussian	Symmetric spaces	✗	✗
Entropy-maximizing Gaussian	Riemannian manifold	✓	✗
von Mises-Fisher	S^{d-1}, $\mathbb{H}^d$, $V_k(\mathbb{R}^d)$, $SO(d)$	✗	✓
Fisher-Bingham, Kent	S^{d-1}	✓	✗
Langevin-Bingham	$V_k(\mathbb{R}^d)$, $\mathrm{Gr}(d,k)$	–	✗
Souriau-Gibbs, Poincaré	$\mathbb{H}^d$	–	✓
Wishart, t-Wishart	$\mathcal{P}_d$	–	✓
Riemannian Laplace	$\mathcal{P}_d$	✗	✗
Wrapped distributions	S^1, $\mathbb{H}^d$, Lie groups, hom. & sym. manifolds, $\mathcal{P}_d$	✓	✓*

* A closed-form density is not available in general (it requires the Jacobian of the exponential map), but exists for wrapped distributions on specific manifolds such as $\mathbb{H}^d$ and $\mathcal{P}_d$ (see Chapter 3).

Table 2.3. Summary of probability distributions on Riemannian manifolds reviewed in Section 2.3.4. *Anisotropy*: ✓ anisotropic, ✗ isotropic, – not a defining property. *Closed-form*: whether the density can be pointwise evaluated without Monte Carlo approximation.

Wrapped distributions have also been studied on more general manifolds in Galaz-Garcia et al. (2022), in which the authors study them on homogeneous Riemannian manifolds, and use a volume preserving map from the tangent space to the manifold, leading to a simpler density expression (but constructing such a volume preserving map is non-trivial in a general setting). Emmanuel Chevallier has studied wrapped distributions on general symmetric spaces in E. Chevallier, D. Li, et al. (2022) and in E. Chevallier and Guigui (2020). Their study focuses mainly on the derivation of the Jacobian determinant of the exponential map, as we will see that it is a crucial ingredient in Theorem 3.3. Finally, the authors of Troshin and Niculae (2023) defined wrapped β-Gaussian on general Riemannian manifolds.

Wrapped distributions have been used in different applications, such as differential privacy on manifolds to build an exponential mechanism (Y. Jiang et al. 2025) or to build Wrapped Gaussian Process Latent Variable Models (González-Duque et al. 2025). They are also used to build deep generative models representing scRNA-seq data in hyperspherical or hyperbolic spaces (Ding and Regev 2021), or to perform process regression (Mallasto and Feragen 2018).

2.3.4.5 Summary of the SOTA on probability distributions defined on a manifold

In the above section, we have reviewed probability distributions defined on manifolds, some of them only defined on a specific manifold, such as the Wishart on $\mathcal{P}_d$ (Wishart 1928) or the Souriau-Gibbs distribution (Barbaresco 2019) defined on $\mathbb{H}_d$, and some of them, defined on a general class of spaces (such as the isotropic Gaussian from Said, Hajri, et al. 2018, defined on symmetric spaces), or that could be generalized to multiple spaces (such as the von Mises-Fisher distribution R. A. Fisher 1953; Mardia and Jupp 2009).

In Chapter 3, we will build and study a wrapped Gaussian on the manifold $\mathcal{P}_d$ of SPD matrices. The key goal of this work was to build an anisotropic distribution that could be used in practice. This means that one should be able to evaluate the density at a given point in closed-form. We can already study those two properties (anisotropy and closed-form density) for the distributions reviewed in this section. We propose such a structured overview in Table 2.3. One can see that the wrapped Gaussian is the only distribution to fulfill the two properties that we want. Indeed, most of the previously studied distributions on the manifold

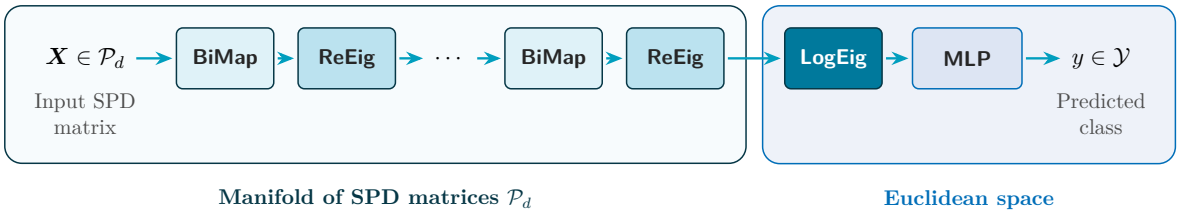


Figure 2.11. Illustration of the SPDNet proposed in Huang and Van Gool (2017). This Riemannian neural network takes as input an SPD matrix $\boldsymbol{X}$, transforms it to other SPD matrices through the BiMap and ReEig layers, and then sends it to a tangent space using the LogEig layer. Finally, any Euclidean MLP can be applied in the Euclidean tangent space in order to predict the output y.

of SPD matrices are isotropic and the entropy-maximizing Gaussian defined by Pennec (2006) requires the evaluation of an integral over the whole manifold to evaluate its density (see Equation (2.11)).

2.3.5 Deep learning on manifolds

This last section of this literature review will be focused on deep learning techniques and models used when the data is not Euclidean but rather lives on a manifold. As for the other methods reviewed above, classical Euclidean deep learning methods need to be adapted in order to take fully into account the geometry of the input space.

When the data lies on the manifold $\mathcal{P}_d$ of symmetric positive definite matrices, the first approach to adapting neural networks to this manifold was the *SPDNet* proposed in Huang and Van Gool (2017). This fully Riemannian network is based on two main layers, the *BiMap* and the *ReEig*. The BiMap is a linear layer and acts as a fully connected layer in classical networks:

$$\boldsymbol{X}_k = f^{(k)}_{\text{BiMap}}(\boldsymbol{X}_{k-1}; \boldsymbol{W}_k) = \boldsymbol{W}_k \boldsymbol{X}_{k-1} \boldsymbol{W}_k^\top,$$

where $\boldsymbol{X}_{k-1} \in \mathcal{P}_{d_{k-1}}$ is the SPD input matrix of the k-th layer, $\boldsymbol{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ is the (learnable) projection matrix and $\boldsymbol{X}_k$ is the output of the layer. This first transformation is the same as the one used in Harandi et al. (2014) and Horev et al. (2016) to reduce the dimension of the SPD data. The *ReEig* plays the role of an activation function that clamps small eigenvalues to a fixed threshold:

$$\boldsymbol{X}_k = f^{(k)}_{\text{ReEig}}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}_{k-1} \max(\epsilon \boldsymbol{I}_{d_{k-1}}, \boldsymbol{\Sigma}_{k-1}) \boldsymbol{U}_{k-1}^\top,$$

where $\boldsymbol{U}_{k-1}$ and $\boldsymbol{\Sigma}_{k-1}$ are obtained using the eigenvalue decomposition of $\boldsymbol{X}_{k-1}$: $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{U}_{k-1}^\top$. The max operator is applied element-wise between the diagonal elements of the input matrix and the threshold ϵ. The ReEig is meant to play a similar role to the ReLU in traditional neural networks. However, Santos et al. (2026) argue that the ReEig layer "serves mainly a numerical role, preventing loss of positive definiteness due to floating-point errors, rather than introducing the nonlinearity expected from an activation function." Thanks to those two layers, the intermediate transformations of the input matrix are still SPD, enabling non-linear learning on Riemannian manifolds. Huang and Van Gool (2017) also introduce a last layer, the *LogEig*, that sends the output of the network to the Euclidean tangent space $T_{I_d}\mathcal{P}_d$ at I_d, where a classical fully-connected layer can be applied to perform classification:

$$\boldsymbol{X}_k = f^{(k)}_{\text{LogEig}}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}_{k-1} \log(\boldsymbol{\Sigma}_{k-1}) \boldsymbol{U}_{k-1}^\top,$$

where $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{U}_{k-1}^\top$ is an eigenvalue decomposition.

This network has established the template that subsequent architectures have extended and is illustrated in Figure 2.11. A missing component of the SPDNet was batch normalization

(Ioffe and Szegedy 2015), a key component of modern deep architectures. Brooks et al. (2019) introduced a Riemannian batch normalization algorithm for SPDNet that generalizes the Euclidean version, making use of geometric operations on the manifold, notably the Riemannian barycenter (Fréchet mean) and parallel transport. The Riemannian batch normalization systematically gives better classification performance compared with leading methods and remarkable robustness to lack of data. Later, Kobler et al. (2022) proposed an extension of the SPD batch normalization to also normalize the dispersion (and not only the barycenter) of the distribution of latent inputs. They obtained better results on domain adaptation in EEG. Other batch normalization methods for SPD matrices have been proposed, such as Gallet et al. (2026) that leverages a parametric mean that automatically adapts between simple arithmetic and harmonic means, or Z. Chen, Yue Song, Y. Liu, et al. (2024) that propose a unified framework for Riemannian Batch Normalization techniques on Lie groups. In Oswald and Massart (2026), the authors showed that the orthogonality constraints imposed on the weight matrices $\boldsymbol{W}_k$ of the BiMap layers limit the expressivity of these layers. More precisely, for certain activation functions, deep architectures collapse to one-hidden-layer networks. Another problem that has been observed empirically is that deeper SPD networks exhibit a loss of generalization capability, meaning that deeper SPDNets perform worse than SPDNets built with a few layers (see Figure 1 of Rui Wang et al. 2022). To solve this problem, the authors of Rui Wang et al. (2022) proposed a new architecture built on top of the SPDNet, called *DreamNet*, that builds a Stacked Riemannian Autoencoder on the tail of the SPDNet backbone. Shortcut connections are also implemented via element-wise addition in order to improve the reconstruction while avoiding adding too much computational complexity. For the specific application of EEG classification relying on covariance matrices, Ju and Guan (2023) mix the SPDNet with a graph-based approach to capture localized fluctuations, enabling simultaneous analysis in the time-frequency domain. The last layer of the SPDNet, the LogEig, has also been replaced by Multinomial Logistic Regression layers to improve the classification performance, as described at the end of Section 2.3.1. For the specific application of BCI, Darley and Bonnet (2025) added a convolutional layer in front of the SPDNet that takes as input the raw EEG and learns a frequency filter. They then estimate the covariance matrix of the filtered signal, and feed it to a classical SPDNet.

Other deep networks for the manifold of SPD matrices have been studied, such as X. S. Nguyen (2022) that proposes two layers (that closely resemble the BiMap and ReEig) relying on a gyrovector approach for SPD matrices. This gyrovector approach has also been used by X. S. Nguyen, S. Yang, and Histace (2024) to build fully-connected and convolutional layers for SPD neural networks. In R. Chakraborty et al. (2022), the authors propose an extension of convolution neural networks on Riemannian manifolds using weighted Fréchet means as convolutions. Indeed, they argue, (based on A. Goh, Lenglet, et al. 2011), that classical convolutions in vector spaces are equivalent to computing weighted means. They then test their new Riemannian convolution layer on the manifold of SPD matrices as well as on the Grassmann manifold. In a very similar way to the SPDNet paper (Huang and Van Gool 2017), the same authors proposed a deep network for the Grassmann manifold in Huang, J. Wu, et al. (2018) and Huang, Ruiping Wang, Shan, and X. Chen (2015) with an extension of a linear layer (as the BiMap), an activation-type layer (as the ReEig) and projection layers (like the LogEig). The recent success of attention in Euclidean deep learning (Vaswani et al. 2017) has encouraged some researchers to try and extend this mechanism to Riemannian manifolds. In Pan et al. (2022), the authors define an attention module for sequences of SPD matrices, where the similarities between queries and keys are computed using the Log-Euclidean distance. They apply their mechanism to EEG data and Qin et al. (2024) to identify cognitive impairment, a vital task for Alzheimer's disease prevention.

Deep learning has also been brought to other manifolds, such as the hyperbolic space $\mathbb{H}^d$. In Ganea et al. (2018) or Shimizu et al. (2020), the authors extend fully-connected

	Intrinsic methods	Extrinsic methods
Supervised learning	MDM, FgMDM (Barachant, Bonnet, et al. 2010b) Geodesic regression (Thomas Fletcher 2011)	LDA/SVM (Barachant, Bonnet, et al. 2011) Regression (Sabbagh et al. 2019; Mellot, Collas, S. Chevallier, Gramfort, et al. 2024)
Kernel methods	Riemannian Gaussian kernels (Feragen et al. 2015)	Tangent-space kernels (Barachant, Bonnet, et al. 2013) Log-Euclidean kernels (Jayasumana et al. 2013; Wagner vom Berg et al. 2025)
Dimension reduction	PGA (in theory) (P. Fletcher et al. 2004) Geometry-aware PCA (Harandi et al. 2014; Horev et al. 2016) Riemannian LLE (A. Goh et al. 2008) Riemannian t-SNE (Y. Guo et al. 2022; Klimovskaia et al. 2020; Y. Zhou et al. 2021)	PGA (in practice) (Sommer, Lauze, and M. Nielsen 2014) Tangent-space t-SNE (J. Liu et al. 2026)
Probability distributions	Isotropic Gaussian (Said, Hajri, et al. 2018) Entropy Maximizing Gaussian (Pennec 2006) Mises-Fisher distribution (Barndorff-Nielsen 1977; Downs 1972) Wishart distribution (Wishart 1928)	Wrapped distributions (Collett et al. 1981; Nagano et al. 2019; Galaz-Garcia et al. 2022; E. Chevallier, D. Li, et al. 2022)
Deep learning	BiMap and ReEig (Huang et al. 2017) Riemannian BatchNorm (Brooks et al. 2019; Kobler et al. 2022)	LogEig (Huang et al. 2017)

Table 2.4. Overview of machine learning methods for data on a manifold and their classification into two categories: the intrinsic and extrinsic methods.

layers to $\mathbb{H}^d$ relying on different models of the hyperbolic space (Benedetti and Petronio 1992). Specific activation functions for $\mathbb{H}^d$ have also been studied (Q. Liu et al. 2019; Chami, Z. Ying, et al. 2019). Interestingly, in Shimizu et al. (2020), the authors remove the activation functions, arguing that the manifold-valued operations are intrinsically non-linear and therefore make additional activation functions unnecessary. As mentioned in Section 2.3.4, variational autoencoders have been developed for $\mathbb{H}^d$, relying on the wrapped Gaussian (Nagano et al. 2019; Cho et al. 2022; Mathieu et al. 2019). We refer to W. Peng et al. (2021) for a comprehensive review on deep learning on the hyperbolic space $\mathbb{H}^d$. On Lie groups, fully connected and pooling layers have been designed in Huang, Wan, et al. (2017) while an attempt at extending attention is proposed in Fuchs et al. (2020).

2.3.6 Classifying the SOTA methods between intrinsic and extrinsic methods

Now that we have reviewed a large number of ML methods used for learning on a manifold, we provide a high-level analysis of the latter and perform a classification into two types. Indeed, the attentive reader will have noticed that throughout all the different categories described above, supervised learning, kernel methods, dimension reduction algorithms, probability distributions or deep learning methods, one can categorize the different methods into two classes: the *intrinsic methods* and the *extrinsic ones*. The methods of the first category, the intrinsic methods, are defined directly on the manifold and do not rely on any tangent space approximation of the manifold. For the latter category, the methods consist in selecting a well-suited tangent space (often at the Fréchet mean of a given dataset), projecting the points from the manifold to that tangent space and applying a classical Euclidean method. Let us look at the case of supervised learning. In the first category, the intrinsic methods, one has

the MDM (Barachant, Bonnet, et al. 2010b) and the geodesic regression (Thomas Fletcher 2011) for example. On the other hand, the Tangent-Space LDA or SVM (Barachant, Bonnet, et al. 2011) and Tangent-Space regression (Sabbagh et al. 2019; Mellot, Collas, S. Chevallier, Gramfort, et al. 2024) belong to the second category. We have categorized most of the previous SOTA methods in these two categories in Table 2.4.

Relying on a tangent space can have some advantages, as once the data has been mapped to this Euclidean space, any classical Euclidean method can be applied. This is a simple solution when wanting to take into account the known geometry of the data without having to rewrite a fully Riemannian algorithm. Moreover, as we saw in Section 2.2.3.5, the log-Euclidean geometry defined in Definition 2.25 is significantly faster to compute numerically. However, this supposes that the dataset is not too spread out on the manifold, as the whole algorithm will rely on a linear approximation at one specific reference point of the manifold, most of the time the Fréchet mean. If some points are far away from this reference point, this point cloud will get distorted once projected to the tangent space and this could degrade the performances of the Euclidean algorithm used in the tangent space (Jaquier, Rozo, and Asfour 2024). On the other hand, building an intrinsic method will require more work as one cannot simply plug Euclidean methods into the pipeline but will guarantee the geometry of the underlying manifold to be fully respected.

This duality between intrinsic and tangent-space methods will be at the core of Chapter 4 as we will lay some theoretical foundations for both categories. Indeed, we will see that we are able to rewrite most tangent-space methods using the wrapped Gaussian that will be defined in Chapter 3 and that we can build new methods that take the best of both worlds.

Part I

Theory

3

Chapter 3

The wrapped Gaussian distribution

Contents

3.1 Introduction 63
3.2 The wrapped Gaussian on the manifold of SPD matrices 66
3.2.1 The definition of a wrapped Gaussian 66
3.2.2 The density of a wrapped Gaussian 67
3.2.3 Properties of the wrapped Gaussian 71
3.2.4 An equivalence relation between wrapped Gaussians 75
3.2.5 Wrapped Elliptically Contoured Distribution 81
3.3 Estimation of the parameters of a wrapped Gaussian 82
3.3.1 The Maximum Likelihood Estimator 82
3.3.2 Method of moments 85
3.3.3 Numerical experiments 85
3.4 Extension to other manifolds 86
3.4.1 Wrapped Gaussian on a Hadamard manifold 87
3.5 Conclusion 91

3.1 Introduction

Gaussian distributions are ubiquitous in the natural and social sciences to represent real-valued random variables whose distribution is not known. One of the main reasons for this is the *Central Limit Theorem* (CLT), which roughly states that if a random variable X is the sum of many independent random variables, then X will be approximately normally distributed (see Theorem 5.12 of Wasserman 2004, for the exact formulation of the CLT). Therefore, physical quantities that are expected to be the sum of many independent processes, such as measurement errors, can safely be modeled by a Gaussian distribution. Moreover, the Gaussian distribution has some unique theoretical properties that can be valuable in analytic studies. Examples of such properties are its finite moments of all orders, its stability under linear combinations, and its closed-form density. The univariate Gaussian distribution $\mathcal{N}(\mu, \sigma)$, defined on $\mathbb{R}$, is generalized to vectors by the *multivariate* Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ on $\mathbb{R}^d$. In this case, the scaling parameter is a matrix, often denoted Σ, which we call the

dispersion matrix[1]. When $X \sim \mathcal{N}(\mu, \Sigma)$, one can encode in the dispersion matrix Σ specific directions in space, reflecting the relationship between the components of X.

With the goal of modeling the variabilities in BCI in order to answer question 3, the Gaussian distribution could thus be a good choice. Thanks to the dispersion parameter Σ, we could model and study the differences between different mental states. By summarizing the complex information of the EEG into a statistical model like the Gaussian one, the hope is that the important information will be concentrated in the parameters of the model and that focusing on them will help us to better understand what is interesting among the chaos of the data. However, an important hidden assumption, central to applying this ambitious plan, is that the data of interest lie in a flat Euclidean space. As we have discussed in the previous chapters, this is not the setting of this thesis, where the main features we rely on are covariance matrices estimated from EEGs, and where we more broadly study the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. We therefore cannot simply use the classical Euclidean Gaussian, and need to focus on a probability distribution defined on the manifold that respects its curved structure. This is what we aim to do in this chapter: **the goal of this chapter is to define and study a geometry-aware version of the multivariate Gaussian distribution on the manifold $\mathcal{P}_d$ of SPD matrices.** This will give us an answer to question 1.

A well-grounded yet tractable distribution

We have already seen in Section 2.3 that there are different Riemannian Gaussians defined on the manifold of SPD matrices. Specifically, there is the isotropic Gaussian from Said, Bombrun, et al. (2017) and the entropy-maximizing Gaussian from Pennec (2006). Then, why would we bother defining and studying another version of a Riemannian Gaussian? To answer this question, let us list the properties we want our Gaussian to satisfy. We want a distribution that is

- **defined on the manifold of SPD matrices**, as this is our space of interest and because usual Euclidean distributions do not respect its geometry,
- **anisotropic**, in order to define preferred directions,
- **easy to sample from**, because it will let us quickly generate random data points,
- **with closed-form density** that can be easily evaluated, helping the comparison of the likelihoods of different points,
- **whose parameters can be estimated**, in order to use this distribution to model real data whose parameters are unknown.

Thus, we want a well-defined distribution with some interesting mathematical properties, but that must also be usable in practice. That is, we do not want to build a purely theoretical object: we want to be able to sample points from this distribution, to compute its density numerically, and to estimate its parameters. Let us now check whether the isotropic Gaussian or the entropy-maximizing Gaussian satisfy these properties. Both distributions are defined on the manifold of SPD matrices (the entropy-maximizing one is defined regardless of the manifold, so one can define it on $\mathcal{P}_d$). As its name indicates, the isotropic Gaussian **is not** anisotropic whereas the entropy-maximizing one is anisotropic. A non-trivial algorithm exists to sample points from the isotropic Gaussian, based on a rejection-sampling method. For the entropy-maximizing Gaussian, to the best of our knowledge, no sampling algorithm is known. For both these Gaussians, the density relies on a normalizing constant that is only computable as an integral over the whole manifold, making the point-wise evaluation computationally demanding. Finally, the parameter estimation problem has been studied for the isotropic Gaussian, but we are not aware of any similar work for the entropy-maximizing

[1] We do not use the classical name (covariance matrix) in order to avoid confusion between the spatial covariance matrices estimated from the EEG, and the dispersion parameters of the multivariate Gaussian distribution.

Property	Isotropic Gaussian	Entropy-maximizing Gaussian	**Wrapped Gaussian**
Defined on $\mathcal{P}_d$	✓	✓	✓
Anisotropic	✗	✓	✓
Easy to sample	✗	✗	✓
Closed-form density	✗	✗	✓
Parameter estimation	$\sim^\star$	✗	✓

* An MLE is known for the scaling parameter, but solving it in practice is computationally expensive.

Table 3.1. Comparison of the main Riemannian Gaussian distributions against the proposed Wrapped Gaussian.

Gaussian. Moreover, the estimation of the scaling parameter of the isotropic Gaussian relies on solving the non-linear Equation (2.10) that also involves the unknown normalizing constant. We summarize these differences in Table 3.1. In what follows, we will introduce a *wrapped Gaussian* by leveraging the Riemannian exponential map to wrap the Euclidean Gaussian defined on a tangent space onto the manifold itself. We will see that this wrapped Gaussian fulfills all of the properties we require.

Compared to already existing wrapped distributions (see Section 2.3.4.4), we consider a wider definition: instead of forcing the density to be centered in the tangent space before being pushed forward to the manifold, we allow it to have a non-zero mean. This adds a parameter to our distribution, and this added degree of freedom will cause some identifiability issues that will be studied and solved in Section 3.2.4. We will thus show that our model can be restricted to classical models where the Euclidean distribution is centered in the tangent space, but that it is also strictly more general.

Outline of the chapter

We will start by studying the wrapped Gaussian on $\mathcal{P}_d$ from a theoretical point of view in Section 3.2. For this, we will properly define the wrapped Gaussian on $\mathcal{P}_d$ in Section 3.2.1, give its density in Section 3.2.2 and some of its properties in Section 3.2.3. An important focus will be put on the identifiability issue raised by the definition of the wrapped Gaussian in Section 3.2.4, and we will show how one can solve it. Then, we will tackle the parameter estimation problem in Section 3.3, using a *Maximum Likelihood Estimator* (MLE), and conduct some numerical experiments. Finally, in Section 3.4, we will see how one can extend the work done on $\mathcal{P}_d$ to other manifolds, with a focus on Hadamard manifolds.

The study of the wrapped Gaussian and its parameter estimation problem was published in:

> **Thibault de Surrel**, Fabien Lotte, Sylvain Chevallier, and Florian Yger (2025). "Wrapped Gaussian on the manifold of Symmetric Positive Definite Matrices". In: *Forty-second International Conference on Machine Learning*

We propose here an enhanced version, with clearer explanations as well as new illustrations to better grasp the wrapped Gaussian. The extension to Hadamard manifolds is currently under review:

> **Thibault de Surrel**, Sylvain Chevallier, Fabien Lotte, and Florian Yger (2026). "Wrapped Gaussian on Hadamard Manifolds". Under review at *Foundations of Computational Mathematics*

In this chapter, we denote by $D = d(d+1)/2$ the dimension of the Riemannian manifold of $d \times d$ SPD matrices $\mathcal{P}_d$. For this whole study, we consider the AIRM geometry for the manifold $\mathcal{P}_d$, as defined in Section 2.2.3.1.

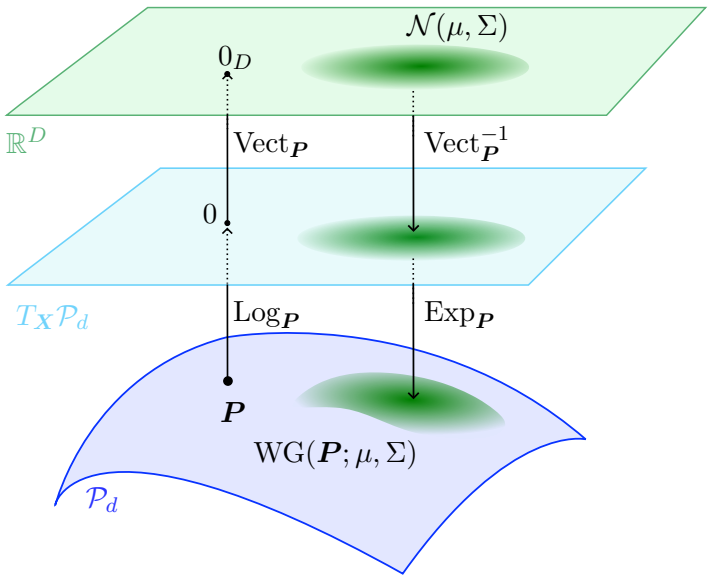


Figure 3.1. Illustration of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ on the manifold of Symmetric Positive Definite matrices $\mathcal{P}_d$. The Euclidean Gaussian $\mathcal{N}(\mu, \Sigma)$ is sent from $\mathbb{R}^D$ to the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ *via* $\mathrm{Vect}_{\boldsymbol{P}}^{-1}$. Then, the Riemannian exponential mapping $\mathrm{Exp}_{\boldsymbol{P}}$ is used to go from $T_{\boldsymbol{P}}\mathcal{P}_d$ to the manifold $\mathcal{P}_d$.

3.2 The wrapped Gaussian on the manifold of SPD matrices

In this section, we give the definition of the wrapped Gaussian, as well as an algorithm to sample from this distribution and some of its theoretical properties.

3.2.1 The definition of a wrapped Gaussian

The *wrapped Gaussian* distribution on the manifold of SPD matrices $\mathcal{P}_d$ is defined by sending a classical Euclidean Gaussian distribution from the tangent space at a point $\boldsymbol{P} \in \mathcal{P}_d$ to the manifold using the exponential map. More formally, we can define the wrapped Gaussian distribution as follows:

Definition 3.1 (Wrapped Gaussian distribution)**.** Let $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ and Σ be a $D \times D$ positive definite matrix. A random variable $\boldsymbol{X}$ on $\mathcal{P}_d$ is said to follow a wrapped Gaussian distribution denoted $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ if

$$\boldsymbol{X} = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t)), \quad t \sim \mathcal{N}(\mu, \Sigma).$$

Therefore, the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ is the push-forward measure (see Section 3.6 of Bogachev 2007) of the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ by the map $\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1} : \mathbb{R}^D \to \mathcal{P}_d$:

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \left(\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}\right) \# \mathcal{N}(\mu, \Sigma), \tag{3.1}$$

where $\#$ denotes the push-forward. The definition of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ gives us a straightforward way to sample from this distribution: one simply needs to sample t from the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ and then compute $X = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t))$. On the manifold of SPD matrices $\mathcal{P}_d$ equipped with the affine-invariant metric, the mappings $\mathrm{Exp}_{\boldsymbol{P}}$ and $\mathrm{Vect}_{\boldsymbol{P}}^{-1}$ have closed-form formulas given in Section 2.2, so points can be efficiently sampled from the wrapped Gaussian distribution. We give an explicit sampling algorithm in Algorithm 1.

The wrapped Gaussian as defined in Definition D.1 depends on three parameters: $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$. The parameter $\boldsymbol{P}$ serves as an anchor point on the manifold $\mathcal{P}_d$, defining the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ from which the Gaussian distribution is wrapped. We will refer to

Algorithm 1: Sampling from a wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$

Input: $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$, $\Sigma \in \mathcal{P}_D$
Output: $\boldsymbol{X} \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$
Sample $t \sim \mathcal{N}(\mu, \Sigma)$;
Compute $\boldsymbol{X} \leftarrow \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t))$;
return $\boldsymbol{X} \in \mathcal{P}_d$;

this parameter $\boldsymbol{P}$ as the *anchor of the wrapped Gaussian*. Then, the two other parameters are the classical parameters of a multivariate Gaussian in $\mathbb{R}^D$ as the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ is isometric to $\mathbb{R}^D$ *via* the vectorization $\mathrm{Vect}_{\boldsymbol{P}}$: μ is the mean of the Gaussian in $\mathbb{R}^D$, therefore $\mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu)$ models how far we deviate from the origin of the tangent space (we can think of the origin of $T_{\boldsymbol{P}}\mathcal{P}_d$ as the point where the tangent space provides the best linear approximation of the manifold $\mathcal{P}_d$ around $\boldsymbol{P}$) and Σ is the dispersion matrix of the Gaussian[2]. We give an illustration of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ on the manifold $\mathcal{P}_d$ in Figure D.2. It should be noted that in other works, such as in Galaz-Garcia et al. (2022), Nagano et al. (2019), and Troshin and Niculae (2023), the authors restrict the wrapped Gaussian to be centered in the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$, therefore enforcing $\mu = 0$. In this work, we allow $\mu \neq 0$, thus extending the expressivity and flexibility of the model. Indeed, on non-flat manifolds, moving away from zero in $T_{\boldsymbol{P}}\mathcal{P}_d$ and then projecting back onto $\mathcal{P}_d$ using the exponential map will not necessarily result in a change of $\boldsymbol{P}$, but can define a new distribution on $\mathcal{P}_d$. We will show more precisely in Section 3.2.4 that not all wrapped Gaussians can be reduced to a wrapped Gaussian with $\mu = 0$.

Remark 3.2. When the data one is trying to model are $d \times d$ spatial covariance matrices estimated from EEGs, as it will be the case in most of this thesis, the $D \times D$ dispersion matrix Σ of the wrapped Gaussian represents the relationships between the coefficients of the spatial covariance matrices. This is why we need a $D \times D = d(d+1)/2 \times d(d+1)/2$ dispersion matrix Σ: there are $d(d+1)/2$ coefficients in the spatial covariance matrices, thus $d(d+1)/2 \times d(d+1)/2$ possible relationships. This can be seen as a fourth-order moment of the EEG.

3.2.2 The density of a wrapped Gaussian

As stated in Section 3.3 of Casella and R. Berger (2024), one of the strengths of the Euclidean Gaussian is that its density is tractable analytically. We now show that this is also the case for the wrapped Gaussian, using its push-forward definition given in Equation (D.1). One is able to derive the density of the distribution $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$:

Theorem 3.3 (Density of the wrapped Gaussian)**.** *The density $f_{\boldsymbol{P};\mu,\Sigma}$ of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ exists with respect to the Riemannian volume of $\mathcal{P}_d$ and is:*

$$\forall \boldsymbol{X} \in \mathcal{P}_d,\ f_{\boldsymbol{P};\mu,\Sigma}(\boldsymbol{X}) = \frac{g_{\mu,\Sigma}(\mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X})))}{|J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}))|}, \tag{3.2}$$

where $g_{\mu,\Sigma}$ is the density of the multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ in $\mathbb{R}^D$, and $J_{\boldsymbol{P}}(\cdot) = \det(\mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\cdot))$ is the Jacobian determinant of the exponential map $\mathrm{Exp}_{\boldsymbol{P}}$.

Proof. Let $T_{\boldsymbol{P}} = \mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1} \colon \mathbb{R}^D \to \mathcal{P}_d$. As $\mathcal{P}_d$ equipped with the AIRM is a Hadamard manifold, $\mathrm{Exp}_{\boldsymbol{P}}$ is a global diffeomorphism with inverse $\mathrm{Log}_{\boldsymbol{P}}$ (see Proposition 2.19), and

[2]We recall that we call Σ the *dispersion matrix* and not the *covariance* matrix to avoid confusion between the covariance matrices of the EEG (the points on $\mathcal{P}_d$) and Σ in the applications.

$\mathrm{Vect}_{\boldsymbol{P}}^{-1}$ is a linear isomorphism; hence $T_{\boldsymbol{P}}$ is a global diffeomorphism, with inverse $T_{\boldsymbol{P}}^{-1} = \mathrm{Vect}_{\boldsymbol{P}} \circ \mathrm{Log}_{\boldsymbol{P}}$.

Let $\varphi\colon \mathcal{P}_d \to \mathbb{R}$ be bounded and measurable. The change of variable formula for push-forward measures (see Theorem 3.6.1 of Bogachev 2007) together with the definition of the wrapped Gaussian given in Equation (D.1) gives

$$\int_{\mathcal{P}_d} \varphi \,\mathrm{dWG}(\boldsymbol{P}; \mu, \Sigma) = \int_{\mathbb{R}^D} \varphi(T_{\boldsymbol{P}}(t))\, g_{\mu,\Sigma}(t)\,\mathrm{d}t.$$

Since $T_{\boldsymbol{P}}$ is a diffeomorphism, the change of variables $\boldsymbol{X} = T_{\boldsymbol{P}}(t)$ between the Lebesgue measure of $\mathbb{R}^D$ and the Riemannian volume of $\mathcal{P}_d$ (see Proposition 2.41 of J. M. Lee 2018) reads $\mathrm{dvol}(\boldsymbol{X}) = |\det \mathrm{d}T_{\boldsymbol{P}}(t)|\,\mathrm{d}t$, so that

$$\int_{\mathcal{P}_d} \varphi \,\mathrm{dWG}(\boldsymbol{P}; \mu, \Sigma) = \int_{\mathcal{P}_d} \varphi(\boldsymbol{X}) \frac{g_{\mu,\Sigma}(T_{\boldsymbol{P}}^{-1}(\boldsymbol{X}))}{|\det \mathrm{d}T_{\boldsymbol{P}}(T_{\boldsymbol{P}}^{-1}(\boldsymbol{X}))|}\,\mathrm{dvol}(\boldsymbol{X}),$$

which shows that $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ is absolutely continuous with respect to the Riemannian volume of $\mathcal{P}_d$.

It remains to identify the Jacobian term. As $\mathrm{Vect}_{\boldsymbol{P}}^{-1}$ is linear, the chain rule gives, for all $t \in \mathbb{R}^D$, $\mathrm{d}T_{\boldsymbol{P}}(t) = \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t)) \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}$. Moreover, $\mathrm{Vect}_{\boldsymbol{P}}^{-1}$ is a linear isometry between $\mathbb{R}^D$ and $T_{\boldsymbol{P}}\mathcal{P}_d$ according to Proposition 2.32, so $|\det \mathrm{Vect}_{\boldsymbol{P}}^{-1}| = 1$ and

$$|\det \mathrm{d}T_{\boldsymbol{P}}(t)| = |\det \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t))| \cdot |\det \mathrm{Vect}_{\boldsymbol{P}}^{-1}| = |J_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t))|.$$

We conclude by noting that $T_{\boldsymbol{P}}^{-1}(\boldsymbol{X}) = \mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}))$ and $\mathrm{Vect}_{\boldsymbol{P}}^{-1}(T_{\boldsymbol{P}}^{-1}(\boldsymbol{X})) = \mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X})$, which yields Equation (3.2). □

We give illustrations of the density of the wrapped Gaussian with different parameters in Figure 3.3.

In this expression, the Jacobian determinant of the exponential map $J_{\boldsymbol{P}}$ is a crucial term that accounts for the change of volume induced by the exponential map $\mathrm{Exp}_{\boldsymbol{P}}$ when wrapping the Gaussian $\mathcal{N}(\mu, \Sigma)$ on $\mathcal{P}_d$. In general, this Jacobian determinant $J_{\boldsymbol{P}}$ is linked to the curvature of the manifold. Fortunately, in the case of the manifold of SPD matrices $\mathcal{P}_d$, we have a closed-form formula for $J_{\boldsymbol{P}}$.

Proposition 3.4 (Jacobian of the exponential map)**.** *The Jacobian determinant of the exponential map* Exp_{I_d} *at the identity is:*

$$\forall \boldsymbol{U} \in T_{I_d}\mathcal{P}_d,\ J_{I_d}(\boldsymbol{U}) = 2^{\frac{d(d-1)}{2}} \prod_{i<j} \frac{\sinh\left(\frac{\lambda_i(\boldsymbol{U}) - \lambda_j(\boldsymbol{U})}{2}\right)}{\lambda_i(\boldsymbol{U}) - \lambda_j(\boldsymbol{U})} \tag{3.3}$$

where the $\lambda_i(\boldsymbol{U})$ *are the eigenvalues of* $\boldsymbol{U}$*. Then, one can use the previous formula to compute the Jacobian determinant of the exponential map at any point* $\boldsymbol{P} \in \mathcal{P}_d$*:*

$$\forall \boldsymbol{U} \in T_{\boldsymbol{P}}\mathcal{P}_d,\ J_{\boldsymbol{P}}(\boldsymbol{U}) = J_{I_d}(\boldsymbol{P}^{-1/2}\boldsymbol{U}\boldsymbol{P}^{-1/2}).$$

Using this, we are able to compute explicitly the density of a point $\boldsymbol{X} \in \mathcal{P}_d$ under the wrapped Gaussian distribution $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$. This will be used in Section 3.3 to estimate the parameters of a wrapped Gaussian given a finite number of samples. The proof of the previous proposition is given in Appendix A.2.1. Let us study how this Jacobian determinant behaves.

Proposition 3.5. *For all* $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d, J_{I_d}(\boldsymbol{U}) \geqslant 1$ *with equality if, and only if, there exists* $t \in \mathbb{R}$ *such that* $\boldsymbol{U} = tI_d$*.*

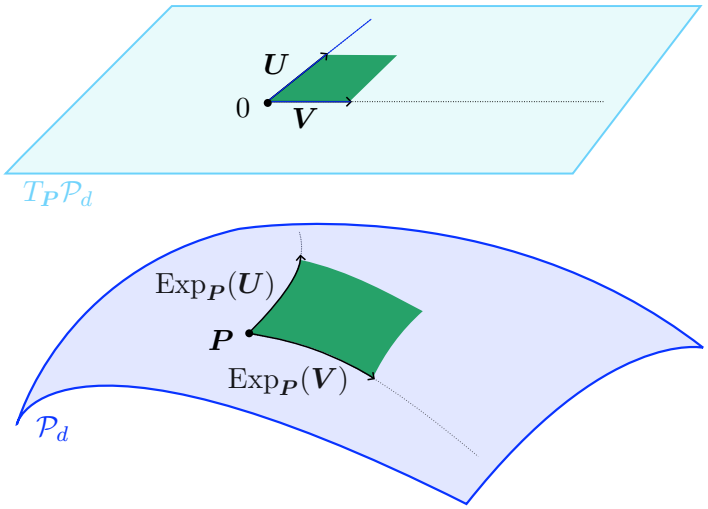


Figure 3.2. Illustration of the Jacobian determinant $J_{\boldsymbol{P}}$ of the Riemannian exponential mapping $\mathrm{Exp}_{\boldsymbol{P}}$. The infinitesimal volume change between the green areas depends on $J_{\boldsymbol{P}}$. On $\mathcal{P}_d$, according to Proposition 3.5, the volumes are increasing from $T_{\boldsymbol{P}}\mathcal{P}_d$ to the manifold $\mathcal{P}_d$. This figure was inspired by figure 1.1 of E. Chevallier, D. Li, et al. (2022).

Proof. Using Proposition 3.4, we recall that, for $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d \cong \mathcal{S}_d$, one has

$$J_{I_d}(\boldsymbol{U}) = 2^{\frac{d(d-1)}{2}} \prod_{i<j} \frac{\sinh\left(\frac{\lambda_i(\boldsymbol{U})-\lambda_j(\boldsymbol{U})}{2}\right)}{\lambda_i(\boldsymbol{U})-\lambda_j(\boldsymbol{U})} = \prod_{i<j} \frac{\sinh\left(\frac{\lambda_i(\boldsymbol{U})-\lambda_j(\boldsymbol{U})}{2}\right)}{\frac{\lambda_i(\boldsymbol{U})-\lambda_j(\boldsymbol{U})}{2}},$$

where the $\lambda_i(\boldsymbol{U})$ are the eigenvalues of $\boldsymbol{U}$. We know that $\frac{\sinh(x)}{x} \geqslant 1$ with equality if, and only if $x = 0$, thus, we have the first result: $J_{I_d}(\boldsymbol{U}) \geqslant 1$. Moreover, let us suppose that $J_{I_d}(\boldsymbol{U}) = 1$. Since each term of the product is greater than 1, this implies that each term is equal to 1 i.e.

$$\forall i < j,\ \lambda_i(\boldsymbol{U}) - \lambda_j(\boldsymbol{U}) = 0 \iff \forall i < j,\ \lambda_i(\boldsymbol{U}) = \lambda_j(\boldsymbol{U}).$$

Thus, all the eigenvalues of $\boldsymbol{U}$ are equal. $\boldsymbol{U}$ being symmetric and real, it is diagonalizable, so it is of the form $\boldsymbol{U} = tI_d$ with $t \in \mathbb{R}$. One can easily check that if $\boldsymbol{U} = tI_d$, then $J_{I_d}(\boldsymbol{U}) = 1$. □

This behavior of the Jacobian determinant of the exponential mapping being greater than one is not surprising. It directly reflects the non-positive sectional curvature inherent to the manifold $\mathcal{P}_d$ equipped with the affine-invariant metric. In spaces with non-positive curvature, geodesics diverge at least as rapidly as they do in flat Euclidean space, meaning the exponential map naturally acts as a local volume expander away from the origin. This phenomenon is illustrated in Figure 3.2.

Remark 3.6. Looking at the expression of the density of a wrapped Gaussian as given in Equation (3.2), one can see that, without any further assumptions, the wrapped Gaussian does not form an exponential family (see Section 9.13.3 of Wasserman 2004). Indeed, the denominator of the density depends both on the parameter $\boldsymbol{P}$ and on the point $\boldsymbol{X}$ of evaluation. It rather forms an exponential family distorted by the mapping from the tangent space.

Remark 3.7. In Section 2.3.4, we have introduced two other Gaussian distributions defined on a Riemannian manifold: the isotropic Gaussian from Said, Bombrun, et al. (2017) and the entropy-maximizing Gaussian from Pennec (2006). One could wonder about the link between the wrapped Gaussian and these two other Gaussians. We have already seen that, when restricted to an isotropic dispersion matrix $\Sigma = \sigma^2 I_D$, the entropy-maximizing

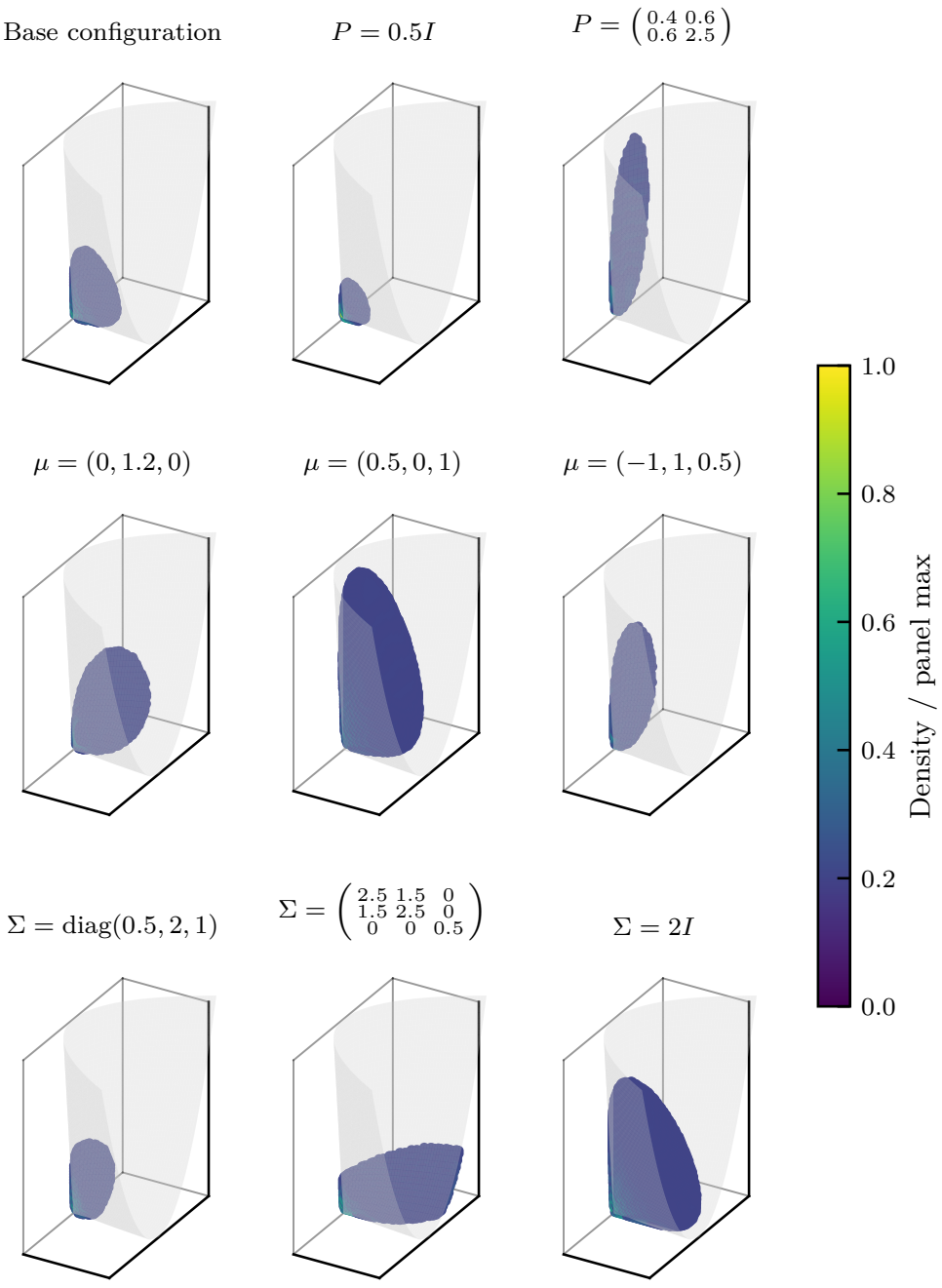


Figure 3.3. Illustration of the density of the wrapped Gaussian distribution $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ for different choices of parameters. The base case is the upper left panel: $(\boldsymbol{P}, \mu, \Sigma) = (I_d, 0_D, I_D)$. Then, each row changes only one parameter (the first row is for $\boldsymbol{P}$, the second for μ and the third for Σ). We do not display the full density (as it would fill the whole cone); instead, we show the regions where the density is the highest, each panel being normalized independently. One can see that each parameter has a specific effect on the distribution, and especially, that the wrapped Gaussian is not isotropic.

Gaussian reduces to the isotropic Gaussian. However, this is not the case for the wrapped Gaussian. There is **no** specific choice of parameters $(\boldsymbol{P}, \mu, \Sigma)$ such that the wrapped Gaussian is equal either to the isotropic one or to the entropy-maximizing one. This is due to the Jacobian determinant of the exponential mapping that is involved in the density of the wrapped Gaussian (Theorem 3.3). Thus, the wrapped Gaussian is a new type of Riemannian Gaussian.

3.2.3 Properties of the wrapped Gaussian

After defining the wrapped Gaussian and giving its density, we are able to give some of its properties, such as a rescaling property, insights into the moments of the wrapped Gaussian and a wrapped version of the Central Limit Theorem.

3.2.3.1 Rescaling property

Let us start with a rescaling property of the wrapped Gaussian. This property states that, by applying some transformations to a wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$, one can obtain another wrapped Gaussian with different parameters.

Proposition 3.8 (Rescaling property)**.** *Let $\boldsymbol{X} \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$. Then, one has the following:*

1. $\boldsymbol{P}^{-1/2}\boldsymbol{X}\boldsymbol{P}^{-1/2} \sim \mathrm{WG}(I_d; \mu, \Sigma)$,
2. $\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu)) \sim \mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$,
3. $\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(\Sigma^{-1/2}\,\mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X}))) \sim \mathrm{WG}(\boldsymbol{P}; \mu, I_D)$.

By combining the three previous points, one obtains an invertible map Ψ such that $\Psi(\boldsymbol{X}) \sim \mathrm{WG}(I_d; 0_D, I_D)$.

This proposition can be used to state that the wrapped Gaussian $\mathrm{WG}(I_d; 0_D, I_D)$ is a building block of the wrapped Gaussian, just like the standard Gaussian $\mathcal{N}(0_d, I_d)$ is a building block of the multivariate Gaussian in $\mathbb{R}^d$. Let us now give the proof.

Proof. In this proof, we will only prove the first two points of the above proposition, the third one being very similar to the second one.

1. Let $\boldsymbol{Y} = \boldsymbol{P}^{-1/2}\boldsymbol{X}\boldsymbol{P}^{-1/2}$. We want to show that $\boldsymbol{Y} \sim \mathrm{WG}(I_d; \mu, \Sigma)$. For this, let $\varphi\colon \mathcal{P}_d \to \mathbb{R}$ be a bounded continuous function. One has
$$\mathbb{E}[\varphi(\boldsymbol{Y})] = \int_{\mathcal{P}_d} \varphi(\boldsymbol{P}^{-1/2}\boldsymbol{X}\boldsymbol{P}^{-1/2}) f_{\boldsymbol{P};\mu,\Sigma}(\boldsymbol{X}) \mathrm{dvol}(\boldsymbol{X}).$$
Let us now define $\psi_{\boldsymbol{P}}\colon \boldsymbol{X} \in \mathcal{P}_d \mapsto \boldsymbol{P}^{-1/2}\boldsymbol{X}\boldsymbol{P}^{-1/2}$. $\psi_{\boldsymbol{P}}$ is a $\mathcal{C}^1$-diffeomorphism between $\mathcal{P}_d$ and $\mathcal{P}_d$. Moreover, as the volume element dvol is invariant under congruence by $\mathrm{GL}(d)$, the transformation $\psi_{\boldsymbol{P}}$ does not induce any volume change. Therefore, by the change of variables $\boldsymbol{Y} = \psi_{\boldsymbol{P}}(\boldsymbol{X})$, one has:
$$\begin{aligned}\mathbb{E}[\varphi(\boldsymbol{Y})] &= \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) f_{\boldsymbol{P};\mu,\Sigma}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y})) \mathrm{dvol}(\boldsymbol{Y}) \\ &= \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) \frac{g_{\mu,\Sigma}(\mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{P}^{1/2}\boldsymbol{Y}\boldsymbol{P}^{1/2})))}{J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{P}^{1/2}\boldsymbol{Y}\boldsymbol{P}^{1/2}))} \mathrm{dvol}(\boldsymbol{Y}).\end{aligned}$$
Now, one can compute that $\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{P}^{1/2}\boldsymbol{Y}\boldsymbol{P}^{1/2}) = \boldsymbol{P}^{1/2}\log(\boldsymbol{Y})\boldsymbol{P}^{1/2}$, thus, using the definition of the vectorization given in Definition 2.31 and the expression of $J_{\boldsymbol{P}}$ given in Proposition 3.4, one has that
$$\begin{aligned}\mathbb{E}[\varphi(\boldsymbol{Y})] &= \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) \frac{g_{\mu,\Sigma}(\mathrm{Vect}_{I_d}(\log(\boldsymbol{Y})))}{J_{I_d}(\log(\boldsymbol{Y}))} \mathrm{dvol}(\boldsymbol{Y}) \\ &= \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) f_{I_d;\mu,\Sigma}(\boldsymbol{Y}) \mathrm{dvol}(\boldsymbol{Y}).\end{aligned}$$

This shows us that $\boldsymbol{Y} \sim \mathrm{WG}(I_d; \mu, \Sigma)$.

2. Let now $\boldsymbol{Y} = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu))$. We want to show that $\boldsymbol{Y} \sim \mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$. Let $\varphi \colon \mathcal{P}_d \to \mathbb{R}$ be a bounded continuous function. One has

$$\mathbb{E}[\varphi(\boldsymbol{Y})] = \int_{\mathcal{P}_d} \varphi(\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu))) f_{\boldsymbol{P};\mu,\Sigma}(\boldsymbol{X}) \mathrm{dvol}(\boldsymbol{X}).$$

Let us now define $\psi_{\boldsymbol{P}} \colon \boldsymbol{X} \mapsto \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu))$. Given the expressions of $\mathrm{Exp}_{\boldsymbol{P}}$ and $\mathrm{Log}_{\boldsymbol{P}}$ (see Proposition 2.19), one has that $\psi_{\boldsymbol{P}}$ is a $\mathcal{C}^1$-diffeomorphism between $\mathcal{P}_d$ and $\mathcal{P}_d$ and its inverse is $\psi_{\boldsymbol{P}}^{-1} \colon \boldsymbol{Y} \mapsto \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{Y} + \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu))$. By the change of variables $\boldsymbol{Y} = \psi_{\boldsymbol{P}}(\boldsymbol{X})$, one has:

$$\begin{aligned}
\mathbb{E}[\varphi(\boldsymbol{Y})] &= \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) \frac{g_{\mu,\Sigma}(\mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y}))))}{J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y})))} \frac{\mathrm{dvol}(\boldsymbol{Y})}{|\det \mathrm{d}\psi_{\boldsymbol{P}}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y}))|} \\
&= \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) \frac{g_{0_D,\Sigma}(\mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y})))}{J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{Y} + \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu))} \frac{\mathrm{dvol}(\boldsymbol{Y})}{|\det \mathrm{d}\psi_{\boldsymbol{P}}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y}))|}
\end{aligned}$$

We need to compute the change of volume term $\det \mathrm{d}\psi_{\boldsymbol{P}}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y}))$. For this, let us start by saying that $\mathrm{d}\psi_{\boldsymbol{P}}(\boldsymbol{X}) = \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu)) \circ \mathrm{d}\,\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X}$ therefore, $\det \mathrm{d}\psi_{\boldsymbol{P}}(\boldsymbol{X}) = \det \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu)) \det \mathrm{d}\,\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X}$. Now using the fact that $\mathrm{d}\,\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y}) = (\mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y})))^{-1}$ and the definition of $J_{\boldsymbol{P}}(\boldsymbol{U}) = \det \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{U})$ (see Theorem 3.3), we have

$$\det \mathrm{d}\psi_{\boldsymbol{P}}(\boldsymbol{X}) = J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X} - \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu)) \frac{1}{J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{X})}$$

and thus, plugging $\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y})$ into this equation:

$$\det \mathrm{d}\psi_{\boldsymbol{P}}(\psi_{\boldsymbol{P}}^{-1}(\boldsymbol{Y})) = J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{Y}) \frac{1}{J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}} \boldsymbol{Y} + \mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu))}.$$

Therefore,

$$\mathbb{E}[\varphi(\boldsymbol{Y})] = \int_{\mathcal{P}_d} \varphi(\boldsymbol{Y}) \frac{g_{0_D,\Sigma}(\mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y})))}{|J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y}))|} \mathrm{dvol}(\boldsymbol{Y}).$$

This shows us that $\boldsymbol{Y} \sim \mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$.

3. For the third point, one can prove it similarly to the two previous ones.

□

3.2.3.2 Moments of the wrapped Gaussian

We also give a result regarding the mean of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ in the special case of $\mu = 0_D$. This special case arises when the Gaussian distribution on the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ is centered. Let us start by recalling from Émery and Mokobodzki (1991) and Pennec (2019), the definition of a mean, or exponential barycenter, of a probability distribution on the Riemannian manifold $\mathcal{P}_d$.

Definition 3.9 (Mean of a distribution on $\mathcal{P}_d$)**.** Let α be a probability distribution on the Riemannian manifold $\mathcal{P}_d$. A mean (or exponential barycenter) of the distribution α is a point $\bar{\boldsymbol{P}} \in \mathcal{P}_d$ such that

$$\int_{\mathcal{P}_d} \mathrm{Log}_{\bar{\boldsymbol{P}}}(\boldsymbol{X}) \mathrm{d}\alpha(\boldsymbol{X}) = 0.$$

Then, the distribution α is centered in the tangent space $T_{\bar{\boldsymbol{P}}}\mathcal{P}_d$. For the specific case of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$, we have the following result:

Proposition 3.10. *A mean of the wrapped Gaussian* $\mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$ *is* $\boldsymbol{P}$.

The proof is straightforward from the definition of the wrapped Gaussian and the fact that the Gaussian distribution $\mathcal{N}(0_D, \Sigma)$ is centered. Using Definition 3 of Pennec (2006), one can also define the variance $\sigma^2_{\boldsymbol{X}}(\boldsymbol{P})$ of a random variable $\boldsymbol{X} \sim \alpha$ on $\mathcal{P}_d$ at a point $\boldsymbol{P} \in \mathcal{P}_d$ by

$$\sigma^2_{\boldsymbol{X}}(\boldsymbol{P}) = \int_{\mathcal{P}_d} \delta(\boldsymbol{P}, \boldsymbol{X})^2 \mathrm{d}\alpha(\boldsymbol{X}).$$

One can compute the variance of a wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ at the "anchor point" $\boldsymbol{P}$.

Proposition 3.11. *Let* $\boldsymbol{X} \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$. *Then, one has* $\sigma^2_{\boldsymbol{X}}(\boldsymbol{P}) = \mathrm{tr}(\Sigma) + \|\mu\|_2^2$.

Proof. Using the definition of the variance and the fact that $\delta(\boldsymbol{P}, \boldsymbol{X}) = \|\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X})\|_{\boldsymbol{P}} = \|\operatorname{Vect}_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}))\|_2$ (the second equality comes from Proposition 2.32), one can apply the change of variables $t = \operatorname{Vect}_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}))$ from $\mathcal{P}_d$ to $\mathbb{R}^D$:

$$\sigma^2_{\boldsymbol{X}}(\boldsymbol{P}) = \int_{\mathcal{P}_d} \delta(\boldsymbol{P}, \boldsymbol{X})^2 f_{\boldsymbol{P};\mu,\Sigma}(\boldsymbol{X}) \mathrm{dvol}(\boldsymbol{X}) = \int_{\mathbb{R}^D} \|t\|_2^2 \, g_{\mu,\Sigma}(t) \mathrm{d}t = \mathrm{tr}(\Sigma) + \|\mu\|_2^2.$$

The last equality comes from the second moment of a multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ on $\mathbb{R}^D$. □

In order to have more general results on the mean and variance of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$, one would need to study the distribution $\operatorname{Log}_{\boldsymbol{Q}} \# \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ where $\boldsymbol{Q} \in \mathcal{P}_d$, i.e. the mapping of the wrapped Gaussian on the tangent space $T_{\boldsymbol{Q}}\mathcal{P}_d$ with $\boldsymbol{Q} \neq \boldsymbol{P}$. The mapping of interest is thus

$$\operatorname{Log}_{\boldsymbol{Q}} \circ \operatorname{Exp}_{\boldsymbol{P}}(\boldsymbol{U}) = \boldsymbol{Q}^{1/2} \log(\boldsymbol{Q}^{-1/2} \boldsymbol{P}^{1/2} \exp(\boldsymbol{P}^{-1/2} \boldsymbol{U} \boldsymbol{P}^{-1/2}) \boldsymbol{P}^{1/2} \boldsymbol{Q}^{-1/2}) \boldsymbol{Q}^{1/2}.$$

However, the mapping $\operatorname{Log}_{\boldsymbol{Q}} \circ \operatorname{Exp}_{\boldsymbol{P}}$ is highly non-linear, and we were not able to derive more general moments of the wrapped Gaussian.

Open Question 3.12. Is it possible to compute general moments of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$?

We give another result on the spread of the wrapped Gaussian that will be useful in Chapter 5:

Proposition 3.13 (Asymptotic spread of the wrapped Gaussian)**.** *Let* $\boldsymbol{X} \sim \mathrm{WG}(I_d; 0_D, I_D)$, *then*

$$\mathbb{E}[\delta(I_d, \boldsymbol{X})] = \sqrt{2} \, \frac{\Gamma\left(\frac{d(d+1)+2}{4}\right)}{\Gamma\left(\frac{d(d+1)}{4}\right)} \sim \sqrt{\frac{d(d+1)}{2}} \quad (d \to +\infty),$$

where Γ *is the Gamma function (see Equation 16 of Davis 1959).*

In order to prove this, let us start with the Euclidean case.

Proposition 3.14. *Let* $X \sim \mathcal{N}(0_d, I_d)$, *then*

$$\mathbb{E}[\|X\|_2] = \sqrt{2} \, \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \sim \sqrt{d} \quad (d \to +\infty).$$

Proof. As $X \sim \mathcal{N}(0_d, I_d)$, its norm $\|X\|_2$ follows a $\chi(d)$ distribution. According to Section 11.3 of M. G. A. Evans et al. (2001), its expectation is

$$\mathbb{E}[\|X\|_2] = \sqrt{2} \, \frac{\Gamma\left(\frac{d+1}{2}\right)}{\Gamma\left(\frac{d}{2}\right)}.$$

Moreover, using Stirling's formula, one gets, according to formula 21 of Davis (1959) that

$$\Gamma(x) \sim x^{x-1/2}e^{-x}\sqrt{2\pi}.$$

Thus, denoting $n = \frac{d}{2}$, one gets

$$\mathbb{E}[\|X\|_2] \sim \sqrt{2}\frac{(n+\frac{1}{2})^n e^{-(n+\frac{1}{2})}\sqrt{2\pi}}{n^{n-\frac{1}{2}}e^{-n}\sqrt{2\pi}} = \sqrt{2}e^{-1/2}\sqrt{n}\left(1+\frac{1}{2n}\right)^n \sim \sqrt{2}\sqrt{n} = \sqrt{d}.$$

□

Going back to the wrapped Gaussian case:

Proof of Proposition 3.13. Let $\boldsymbol{X} \sim \mathrm{WG}(I_d; 0_D, I_D)$. Then, as in the proof of Proposition 3.11,

$$\mathbb{E}[\delta(I_d, \boldsymbol{X})] = \int_{\mathcal{P}_d} \delta(I_d, \boldsymbol{X}) f_{(I_d,0_D,I_D)}(\boldsymbol{X})\mathrm{dvol}(\boldsymbol{X}) = \int_{\mathbb{R}^D} \|t\|_2 \; g_{0_D,I_D}(t)\mathrm{d}t = \mathbb{E}[\|t\|_2],$$

where $f_{(I_d,0_D,I_D)}$ is the density of the wrapped Gaussian $\mathrm{WG}(I_d; 0_D, I_D)$ given in Theorem 3.3, $t \sim \mathcal{N}(0_D, I_D)$ and g_{0_D,I_D} is the density of $\mathcal{N}(0_D, I_D)$. Thus, we can use Proposition 3.14 to conclude:

$$\mathbb{E}[\delta(I_d, \boldsymbol{X})] \sim \sqrt{D} = \sqrt{\frac{d(d+1)}{2}}.$$

□

3.2.3.3 A wrapped Central Limit Theorem

Finally, we give a wrapped version of the multivariate *Central Limit Theorem* (CLT) (Theorem 5.12 of Wasserman 2004) for the manifold $\mathcal{P}_d$. For this, we define the logarithmic product introduced in Arsigny et al. (2006a):

Definition 3.15 (Logarithmic product)**.** Let $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{P}_d$. The logarithmic product of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as:

$$\boldsymbol{X} \odot \boldsymbol{Y} = \exp\left(\log \boldsymbol{X} + \log \boldsymbol{Y}\right).$$

Equipped with this logarithmic product, $(\mathcal{P}_d, \odot)$ forms a commutative group that is isomorphic to $(\mathcal{S}_d, +)$. More information on the properties of this logarithmic product can be found in Arsigny et al. (2006a). One can generalize this notation to the sum of N SPD matrices $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ as (the logarithmic product is associative)

$$\bigodot_{i=1}^{N} \boldsymbol{X}_i = \boldsymbol{X}_1 \odot \cdots \odot \boldsymbol{X}_N.$$

Using this logarithmic product, we can state the following theorem:

Theorem 3.16 (Wrapped CLT)**.** *Let $(\boldsymbol{X}_i)_{i\in\mathbb{N}^*}$ be a sequence of independent and identically distributed random variables on $\mathcal{P}_d$. We suppose that the sequence $(\mathrm{Vect}_{I_d}(\mathrm{Log}_{I_d}(\boldsymbol{X}_i)))_{i\in\mathbb{N}^*}$ of random variables on $\mathbb{R}^D$ admits a finite second order moment. We denote by μ and Σ the first and second moments of $\mathrm{Vect}_{I_d}(\mathrm{Log}_{I_d}(\boldsymbol{X}_1))$. Then,*

$$\left(\bigodot_{i=1}^{N} \boldsymbol{X}_i \odot m^{-1}\right)^{\frac{1}{\sqrt{N}}} \xrightarrow[N\to\infty]{d} \mathrm{WG}(I_d; 0, \Sigma), \tag{3.4}$$

where $\xrightarrow[N\to\infty]{d}$ denotes the convergence in distribution and where $m = \mathrm{Exp}_{I_d}(\mathrm{Vect}_{I_d}^{-1}(\mu))$.

This theorem highlights the relevance of wrapped Gaussians, as they naturally appear in the limit of a logarithmic product of random SPD matrices.

Proof. The sequence $(\boldsymbol{X}_i)_{i\in\mathbb{N}^*}$ is independent and identically distributed (i.i.d.) on $\mathcal{P}_d$. Thus, applying the measurable mapping $\mathrm{Vect}_{I_d}\circ\mathrm{Log}_{I_d}$, the sequence $(\mathrm{Vect}_{I_d}(\mathrm{Log}_{I_d}(\boldsymbol{X}_i)))_{i\in\mathbb{N}^*}$ is still i.i.d. in $\mathbb{R}^D$. As the sequence $(\mathrm{Vect}_{I_d}(\mathrm{Log}_{I_d}(\boldsymbol{X}_i)))_{i\in\mathbb{N}^*}$ satisfies the assumptions of the classical Central Limit Theorem in $\mathbb{R}^D$, one has that,

$$\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\Big(\mathrm{Vect}_{I_d}(\mathrm{Log}_{I_d}(\boldsymbol{X}_i))-\mu\Big)\xrightarrow[N\to\infty]{d}\mathcal{N}(0,\Sigma).$$

By defining $m=\mathrm{Exp}_{I_d}(\mathrm{Vect}_{I_d}^{-1}(\mu))\in\mathcal{P}_d$, and using the linearity of Vect_{I_d}:

$$\mathrm{Vect}_{I_d}\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N}(\mathrm{Log}_{I_d}\boldsymbol{X}_i-\mathrm{Log}_{I_d}m)\right)\xrightarrow[N\to\infty]{d}\mathcal{N}(0,\Sigma).$$

Thus, by applying the continuous map $\mathrm{Exp}_{I_d}\circ\mathrm{Vect}_{I_d}^{-1}$ to the previous equation, by considering the fact that the convergence in distribution is preserved under continuous maps (Theorem 5.5 of Wasserman 2004) as well as the definition of the wrapped Gaussian (Definition D.1) and the expressions of Exp_{I_d} and Log_{I_d} (see Proposition 2.19), one has that

$$\exp\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N}(\log\boldsymbol{X}_i-\log m)\right)\xrightarrow[N\to\infty]{d}\mathrm{WG}(I_d;0,\Sigma).$$

We can now simplify the left-hand side of the previous equation:

$$\begin{aligned}\exp\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N}(\log\boldsymbol{X}_i-\log m)\right)&=\exp\left(\sum_{i=1}^{N}(\log\boldsymbol{X}_i+\log m^{-1})\right)^{\frac{1}{\sqrt{N}}}\\&\qquad\text{(using }\log m=-\log m^{-1}\text{ and }\exp(\alpha x)=\exp(x)^{\alpha}\text{ for }\alpha\in\mathbb{R})\\&=\exp\left(\sum_{i=1}^{N}\log(\boldsymbol{X}_i\odot m^{-1})\right)^{\frac{1}{\sqrt{N}}}\\&\qquad\text{(using the definition of the logarithmic product)}\\&=\left(\bigodot_{i=1}^{N}\boldsymbol{X}_i\odot m^{-1}\right)^{\frac{1}{\sqrt{N}}}.\end{aligned}$$

Therefore, one has the final result of the wrapped Central Limit Theorem:

$$\left(\bigodot_{i=1}^{N}\boldsymbol{X}_i\odot m^{-1}\right)^{\frac{1}{\sqrt{N}}}\xrightarrow[N\to\infty]{d}\mathrm{WG}(I_d;0,\Sigma).$$

□

3.2.4 An equivalence relation between wrapped Gaussians

A key property for a statistical model is its *identifiability*. Let us recall the definition.

Definition 3.17 (Identifiability of a statistical model)**.** Let $\{\mathbb{P}_\theta,\theta\in\Theta\}$ be a statistical model. The model is said to be *identifiable* if $\theta\mapsto\mathbb{P}_\theta$ is injective, i.e. $\theta_1\neq\theta_2\Rightarrow\mathbb{P}_{\theta_1}\neq\mathbb{P}_{\theta_2}$.

This property is satisfied by most classical Euclidean distributions, such as the Gaussian, the exponential, the χ^2, or the Student's t-distributions. Identifiability is particularly important in parameter estimation, as it ensures that distinct parameter values correspond to distinct probability distributions of the observations, making the parameters theoretically recoverable from sufficiently large amounts of data. The problem of parameter estimation for the wrapped

Gaussian on the manifold $\mathcal{P}_d$ will be tackled in Section 3.3. Let us thus first study the identifiability of the wrapped Gaussian. Because of the intrinsic geometry of $\mathcal{P}_d$ equipped with the affine-invariant metric, we will show that the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ defined on the manifold $\mathcal{P}_d$ of SPD matrices is not identifiable. Indeed, several sets of parameters define the same wrapped Gaussian. Let us detail this statement in the following theorem:

Theorem 3.18 (Equivalent wrapped Gaussians)**.** *Let $\boldsymbol{P}, \boldsymbol{Q} \in \mathcal{P}_d$, $\mu, \nu \in \mathbb{R}^D$ and $\Sigma, \Gamma \in \mathcal{P}_D$. Then, one has the following equivalence:*

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \mathrm{WG}(\boldsymbol{Q}; \nu, \Gamma) \iff \exists t \in \mathbb{R},\ \boldsymbol{Q} = e^t \boldsymbol{P}, \nu = \mu - t\ \mathrm{Vect}_{I_d}(I_d), \Sigma = \Gamma.$$

This gives us a direct corollary on the identifiability of the wrapped Gaussian on $\mathcal{P}_d$.

Corollary 3.19. *The wrapped Gaussian defined on the manifold $\mathcal{P}_d$ is* not *identifiable.*

Proof. Using Theorem 3.18, and let $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$, one has that

$$\forall t \in \mathbb{R},\ \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \mathrm{WG}(e^t \boldsymbol{P}; \mu - t\, \mathrm{Vect}_{I_d}(I_d), \Sigma).$$

As for all $t \neq 1$, $\boldsymbol{P} \neq e^t \boldsymbol{P}$, one has the result. □

Figure 3.4 illustrates Theorem 3.18. Before proving this result, we make a few remarks. Using Proposition 2.19, one can verify that the curve $t \mapsto e^t \boldsymbol{P}$ is precisely the geodesic $\gamma \colon t \mapsto \mathrm{Exp}_{\boldsymbol{P}}(t\boldsymbol{P})$ in $\mathcal{P}_d$, starting at $\boldsymbol{P}$ with initial velocity $\boldsymbol{P}$, where $\boldsymbol{P}$ is viewed as a tangent vector via the identification $T_{\boldsymbol{P}}\mathcal{P}_d \cong \mathcal{S}_d$. On the other hand, the curve $t \mapsto \mu - t\, \mathrm{Vect}_{I_d}(I_d)$ is the geodesic in $\mathbb{R}^D$ starting at μ with initial velocity $-\mathrm{Vect}_{\boldsymbol{P}}(\boldsymbol{P})$, using the fact that $\mathrm{Vect}_{I_d}(I_d) = \mathrm{Vect}_{\boldsymbol{P}}(\boldsymbol{P})$. Therefore, the geodesic on $\mathcal{P}_d$ departing from $\boldsymbol{P}$ with initial velocity $\boldsymbol{P}$ corresponds, under $\mathrm{Vect}_{\boldsymbol{P}}$, to the geodesic on $\mathbb{R}^D$ departing from μ with the opposite initial velocity $-\mathrm{Vect}_{\boldsymbol{P}}(\boldsymbol{P})$. Finally, we are able to compute $\mathrm{Vect}_{I_d}(I_d)$:

$$\mathrm{Vect}_{I_d}(I_d) = \mathrm{Vect}_{\boldsymbol{P}}(\boldsymbol{P}) = (\underbrace{1, \ldots, 1}_{d \text{ ones}}, \underbrace{0, \ldots, 0}_{d(d-1)/2 \text{ zeros}}) \in \mathbb{R}^D.$$

In order to prove Theorem 3.18 on the equivalence between wrapped Gaussians, we first need to prove the following two lemmas.

Lemma 3.20. *Let $\boldsymbol{P} \in \mathcal{P}_d$ and $t \in \mathbb{R}$. Let us define $\tau_t \colon s \in \mathbb{R}^D \mapsto s + tV$, where $V = \mathrm{Vect}_{I_d}(I_d)$. Then,*

$$\mathrm{Exp}_{e^t \boldsymbol{P}} \circ \mathrm{Vect}^{-1}_{e^t \boldsymbol{P}} = \left(\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}^{-1}_{\boldsymbol{P}}\right) \circ \tau_t.$$

Thus, for all $\mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$,

$$\mathrm{WG}(e^t \boldsymbol{P}; \mu, \Sigma) = \mathrm{WG}(\boldsymbol{P}; \mu + tV, \Sigma).$$

Proof. Let $s \in \mathbb{R}^D$. One can compute using Proposition 2.33, that

$$\mathrm{Vect}^{-1}_{e^t \boldsymbol{P}}(s) = (e^t \boldsymbol{P})^{1/2}\, \mathrm{Vect}^{-1}_{I_d}(s) (e^t \boldsymbol{P})^{1/2} = e^t\, \mathrm{Vect}^{-1}_{\boldsymbol{P}}(s).$$

Thus, using the expression of the AIRM exponential mapping given in Proposition 2.19, one has

$$\mathrm{Exp}_{e^t \boldsymbol{P}} \circ \mathrm{Vect}^{-1}_{e^t \boldsymbol{P}}(s) = \mathrm{Exp}_{e^t \boldsymbol{P}}(e^t\, \mathrm{Vect}^{-1}_{\boldsymbol{P}}(s)) = e^t\, \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}^{-1}_{\boldsymbol{P}}(s)).$$

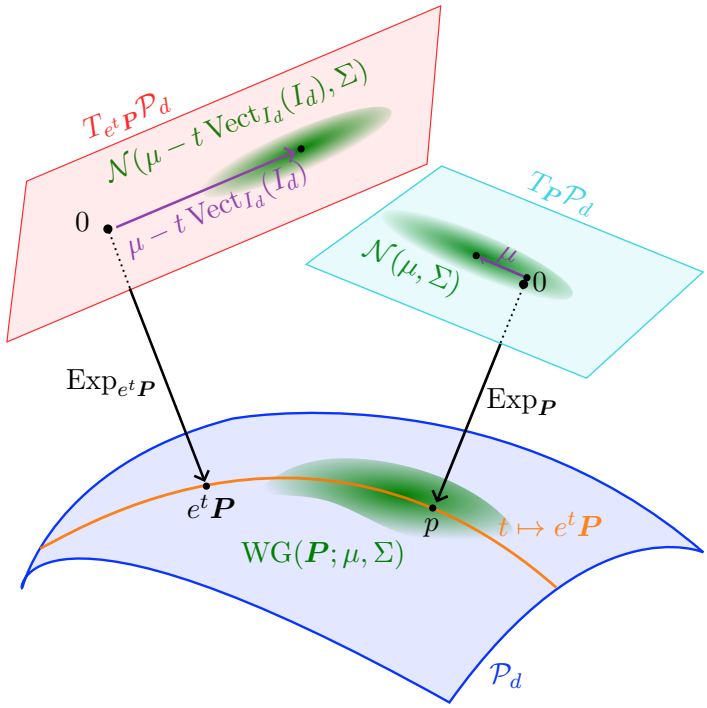


Figure 3.4. Illustration of the equivalence between two wrapped Gaussians on $\mathcal{P}_d$ given by Theorem 3.18. The orange curve is the geodesic on which the wrapped Gaussians are equivalent.

Moreover,

$$\begin{aligned}
\left(\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}\right) \circ \tau_t(s) &= \mathrm{Exp}_{\boldsymbol{P}}\left(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(s + tV)\right) \\
&= \mathrm{Exp}_{\boldsymbol{P}}\left(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(s) + t\underbrace{\mathrm{Vect}_{\boldsymbol{P}}^{-1}(V)}_{=\boldsymbol{P}}\right) \\
&= \boldsymbol{P}^{1/2} \exp\left(\boldsymbol{P}^{-1/2}\,\mathrm{Vect}_{\boldsymbol{P}}^{-1}(s)\boldsymbol{P}^{-1/2}\right) \underbrace{\exp\left(t\boldsymbol{P}^{-1/2}\boldsymbol{P}\boldsymbol{P}^{-1/2}\right)}_{=e^t I_d} \boldsymbol{P}^{1/2} \\
&= e^t\,\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}(s).
\end{aligned}$$

Both sides of the asserted identity therefore send s to $e^t\,\mathrm{Exp}_{\boldsymbol{P}}\left(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(s)\right)$. The second assertion follows from Equation (D.1) and $\tau_t \# \mathcal{N}(\mu, \Sigma) = \mathcal{N}(\mu + tV, \Sigma)$. □

Lemma 3.21. *Let $\phi(s) := \log\left(\frac{\sinh(s/2)}{s/2}\right)$ for $s \neq 0$ and $\phi(0) := 0$. Then, ϕ is continuous on $\mathbb{R}$ and for all $m \in \mathbb{R}$, $s \mapsto \phi(s) - \phi(s-m)$ is bounded on $\mathbb{R}$ with*

$$\lim_{s \to +\infty} \left(\phi(s) - \phi(s-m)\right) = +\frac{m}{2}, \qquad \lim_{s \to -\infty} \left(\phi(s) - \phi(s-m)\right) = -\frac{m}{2}.$$

Proof. ϕ is continuous thanks to the properties of sinh. Thus, $s \mapsto \phi(s) - \phi(s-m)$ is continuous on $\mathbb{R}$ for all $m \in \mathbb{R}$. Moreover,

$$\phi(s) = \log\left(\frac{e^{s/2} - e^{-s/2}}{s}\right) = \log\left(\frac{e^{s/2}}{s}(1 - e^{-s})\right) = \frac{s}{2} - \log s + \log(1 - e^{-s}).$$

Thus,

$$\begin{aligned}\phi(s)-\phi(s-m) &= \frac{s}{2}-\log s+\log(1-e^{-s})-\frac{s-m}{2}+\log(s-m)-\log(1-e^{-(s-m)})\\ &= \frac{m}{2}-\log\left(\frac{s}{s-m}\right)-\log\left(\frac{1-e^{-(s-m)}}{1-e^{-s}}\right)\\ &\xrightarrow[s\to+\infty]{} \frac{m}{2}\\ &\xrightarrow[s\to-\infty]{} -\frac{m}{2} \quad \text{using that } \phi \text{ is even.}\end{aligned}$$

□

Let us now dive into the proof of Theorem 3.18.

Proof of Theorem 3.18. Let us start by the sufficient condition.

$\boxed{\Leftarrow}$ Suppose $\boldsymbol{Q}=e^t\boldsymbol{P}, \nu=\mu-tV$ and $\Gamma=\Sigma$. By Lemma 3.20, one has

$$\mathrm{WG}(\boldsymbol{Q};\nu,\Gamma)=\mathrm{WG}(e^t\boldsymbol{P};\mu-tV,\Sigma)=\mathrm{WG}(\boldsymbol{P};(\mu-tV)+tV,\Sigma)=\mathrm{WG}(\boldsymbol{P};\mu,\Sigma).$$

$\boxed{\Rightarrow}$ Let us now give the proof of the necessary condition. In this first part, we are going to show that if $\mathrm{WG}(\boldsymbol{P};\mu,\Sigma)=\mathrm{WG}(\boldsymbol{Q};\nu,\Gamma)$, then there exists $t\in\mathbb{R}$ such that $\boldsymbol{Q}=e^t\boldsymbol{P}$.

Step 1: Normalization. Using the first point from Proposition 3.8, one can show that it is sufficient to show the above result for $\boldsymbol{P}=I_d$. Thus, in the following, we assume that

$$\mathrm{WG}(I_d;\mu,\Sigma)=\mathrm{WG}(\boldsymbol{Q};\nu,\Gamma) \tag{3.5}$$

Step 2: The log-density identity. As the wrapped Gaussian admits a density with respect to the Riemannian volume according to Theorem 3.3, Equation (3.5) is equivalent to the equality of the densities. These densities can be rewritten as

$$\forall \boldsymbol{U}\in T_{I_d}\mathcal{P}_d,\ \frac{g_{\mu,\Sigma}(\mathrm{Vect}(\boldsymbol{U}))}{|J(\boldsymbol{U})|}=\frac{g_{\nu,\Gamma}(\mathrm{Vect}_{\boldsymbol{Q}}(\mathrm{Log}_{\boldsymbol{Q}}(\mathrm{Exp}_{I_d}(\boldsymbol{U}))))}{|J_{\boldsymbol{Q}}(\mathrm{Log}_{\boldsymbol{Q}}(\mathrm{Exp}_{I_d}(\boldsymbol{U})))|},$$

where we denote $\mathrm{Vect}=\mathrm{Vect}_{I_d}$ and $J=J_{I_d}$. Then, using the expressions of $\mathrm{Exp}_{I_d}, \mathrm{Log}_{\boldsymbol{Q}}$ and $\mathrm{Vect}_{\boldsymbol{Q}}$ given in Proposition 2.19 and Proposition 2.33, taking the logarithm of the above equality simplifies to the following condition for all $\boldsymbol{U}\in T_{I_d}\mathcal{P}_d\simeq\mathcal{S}_d$:

$$\begin{aligned}\frac{1}{2}\|\Sigma^{-1/2}\left(\mathrm{Vect}(\boldsymbol{U})-\mu\right)\|_2^2+\log(J(\boldsymbol{U}))+\frac{1}{2}\log(\det\Sigma)=&\\ \frac{1}{2}\left\|\Gamma^{-1/2}\left(\mathrm{Vect}(\log(\boldsymbol{Q}^{-1/2}\exp(\boldsymbol{U})\boldsymbol{Q}^{-1/2}))-\nu\right)\right\|_2^2&\\ +\log\left(J(\log(\boldsymbol{Q}^{-1/2}\exp(\boldsymbol{U})\boldsymbol{Q}^{-1/2}))\right)+\frac{1}{2}\log(\det\Gamma).&\end{aligned} \tag{3.6}$$

Let us write (thanks to Theorem 2.2), $\boldsymbol{Q}=\boldsymbol{R}\,\mathrm{diag}(e^{q_1},\dots,e^{q_d})\boldsymbol{R}^\top$, where $\boldsymbol{R}\in\mathrm{O}(d)$. In particular, for

$$\boldsymbol{U}_\lambda=\boldsymbol{R}\,\mathrm{diag}(\lambda_1,\dots,\lambda_d)\boldsymbol{R}^\top\in T_{I_d}\mathcal{P}_d,\quad \lambda\in\mathbb{R}^d,$$

one can compute

$$\begin{aligned}\log(\boldsymbol{Q}^{-1/2}\exp(\boldsymbol{U}_\lambda)\boldsymbol{Q}^{-1/2})&=\boldsymbol{U}_\lambda-\log\boldsymbol{Q},\\ \log J(\boldsymbol{U}_\lambda)&=\sum_{i<j}\phi(\lambda_i-\lambda_j),\\ \log J(\log(\boldsymbol{Q}^{-1/2}\exp(\boldsymbol{U}_\lambda)\boldsymbol{Q}^{-1/2}))&=\sum_{i<j}\phi((\lambda_i-\lambda_j)-(q_i-q_j)).\end{aligned}$$

Step 3: The two Jacobian terms. Plugging all this into Equation (3.6) and gathering the Gaussian terms on one side and the Jacobian terms on the other gives us, for all $\lambda \in \mathbb{R}^d$:

$$A(\lambda) = B(\lambda), \tag{3.7}$$

where we have set

$$A(\lambda) := \frac{1}{2}\|\Gamma^{-1/2}(\operatorname{Vect}(\boldsymbol{U}_\lambda) - \operatorname{Vect}(\log \boldsymbol{Q}) - \nu)\|_2^2 - \frac{1}{2}\|\Sigma^{-1/2}(\operatorname{Vect}(\boldsymbol{U}_\lambda) - \mu)\|_2^2 + \frac{1}{2}\log\frac{\det\Gamma}{\det\Sigma}$$

and

$$B(\lambda) := \log(J(\boldsymbol{U}_\lambda)) - \log(J(\boldsymbol{U}_\lambda - \log \boldsymbol{Q})) = \sum_{i<j} \phi(\lambda_i - \lambda_j) - \phi((\lambda_i - \lambda_j) - (q_i - q_j)). \tag{3.8}$$

The left-hand side A is a polynomial function of λ. Moreover, using Lemma 3.21, one has that

$$|B(\lambda)| \leqslant \sum_{i<j} |\phi(\lambda_i - \lambda_j) - \phi((\lambda_i - \lambda_j) - (q_i - q_j))|,$$

and is thus bounded. Therefore, A is a bounded polynomial, so A and B are constant. Two evaluations of B then give the result. First, at $\lambda = 0$, using the definition of B given in Equation (3.8):

$$B(0) = -\log(J(\log \boldsymbol{Q})).$$

Then, using the limits obtained in Lemma 3.21, one can show that

$$\begin{aligned} B(\lambda) &\xrightarrow[\lambda\to+\infty]{} \frac{1}{2}\sum_{i<j}(q_i - q_j) \\ &\xrightarrow[\lambda\to-\infty]{} -\frac{1}{2}\sum_{i<j}(q_i - q_j). \end{aligned}$$

Thus, as B is constant, those limits are equal, hence both equal to zero. Thus, $B \equiv 0$. This gives us

$$\log(J(\log \boldsymbol{Q})) = 0,$$

and using the equality case of Proposition 3.5, this gives $\log \boldsymbol{Q} = tI_d$ for some $t \in \mathbb{R}$ i.e. $\boldsymbol{Q} = e^t I_d$.

Step 4: Conclusion. Using Lemma 3.20, we now have:

$$(\operatorname{Exp}_{I_d} \circ \operatorname{Vect}_{I_d}^{-1})\#\mathcal{N}(\mu, \Sigma) = \mathrm{WG}(I_d; \mu, \Sigma) = \mathrm{WG}(\boldsymbol{Q}; \nu, \Gamma) = (\operatorname{Exp}_{I_d} \circ \operatorname{Vect}_{I_d}^{-1})\#\mathcal{N}(\nu + tV, \Gamma).$$

By the Cartan-Hadamard theorem (see Theorem 12.8 of J. M. Lee 2018), $\operatorname{Exp}_{I_d} \circ \operatorname{Vect}_{I_d}^{-1}$ is a diffeomorphism from $\mathbb{R}^D$ onto $\mathcal{P}_d$; pushing the above equality forward through its inverse gives $\mathcal{N}(\mu, \Sigma) = \mathcal{N}(\nu + tV, \Gamma)$. The multivariate Gaussian model on $\mathbb{R}^D$ being identifiable, we conclude that $\nu + tV = \mu$ and $\Gamma = \Sigma$, i.e.

$$\nu = \mu - t\ \operatorname{Vect}_{I_d}(I_d) \qquad \text{and} \qquad \Gamma = \Sigma.$$

□

Remark 3.22. The equivalence given in Theorem 3.18 is linked to the structure of $\mathcal{P}_d$: one has that $\mathcal{P}_d \cong \mathbb{R} \times \mathcal{SP}_d$ where $\mathcal{SP}_d = \{\boldsymbol{P} \in \mathcal{P}_d \mid \det(\boldsymbol{P}) = 1\}$ is the set of SPD matrices of determinant 1. The diffeomorphism involved is

$$\phi\colon \boldsymbol{P} \in \mathcal{P}_d \mapsto \left(\frac{1}{d}\log\det \boldsymbol{P}, \frac{1}{(\det \boldsymbol{P})^{1/d}}\boldsymbol{P}\right) \in \mathbb{R} \times \mathcal{SP}_d.$$

One can then show that, by denoting $(t, \boldsymbol{Q}) = \phi(\boldsymbol{P})$, one has that

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \mathrm{WG}(\boldsymbol{Q}; \mu + t \operatorname{Vect}_{I_d}(I_d), \Sigma) = \mathrm{WG}(e^{-t}\boldsymbol{P}; \mu + t \operatorname{Vect}_{I_d}(I_d), \Sigma),$$

which is exactly the equivalence given in Theorem 3.18. More details on this decomposition of $\mathcal{P}_d$ can be found in E. Chevallier, Trindade, et al. (2025).

A wrapped Gaussian on $\mathcal{P}_d$ can thus be represented by several sets of parameters, so we define an *equivalence relation* between sets of parameters defining the same wrapped Gaussian:

Definition 3.23 (Equivalence classes of wrapped Gaussians)**.** Let $\theta = (\boldsymbol{P}, \mu, \Sigma) \in \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D$ and $\theta' = (\boldsymbol{P}', \mu', \Sigma') \in \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D$ be two sets of parameters. Then, we say that θ and θ' are equivalent, which we denote by $\theta \sim \theta'$, if they define the same wrapped Gaussian i.e.

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \mathrm{WG}(\boldsymbol{P}'; \mu', \Sigma').$$

We denote by $[\theta]$ the equivalence class of θ:

$$[\theta] = \left\{\theta' = (\boldsymbol{P}', \mu', \Sigma') \mid \theta' \sim \theta\right\}.$$

One can easily verify that the previous relation is, indeed, a well-defined equivalence relation. Using Theorem 3.18, one has the immediate corollary:

Corollary 3.24. *Let $\theta = (\boldsymbol{P}, \mu, \Sigma) \in \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D$. One can describe completely the equivalence class $[\theta]$ of θ:*

$$[\theta] = \left\{(e^t \boldsymbol{P}, \mu - t \operatorname{Vect}_{I_d}(I_d), \Sigma),\ t \in \mathbb{R}\right\}.$$

We are now able to make a very important comment on the wrapped Gaussian that we defined in Definition D.1: our wrapped Gaussian is **strictly more expressive** than variants that impose $\mu = 0$ as in Galaz-Garcia et al. (2022), Nagano et al. (2019), and Troshin and Niculae (2023). Let us make this statement clearer.

Proposition 3.25. *Not every equivalence class $[\theta]$ admits a representative with vanishing mean.*

Proof. By Corollary 3.24, an equivalence class $[(\boldsymbol{P}, \mu, \Sigma)]$ contains a tuple with $\mu' = 0$ if and only if μ is aligned with $\operatorname{Vect}_{I_d}(I_d)$, i.e., there exists $t \in \mathbb{R}$ such that $\mu = t \operatorname{Vect}_{I_d}(I_d)$. Since $\operatorname{Vect}_{I_d}(I_d) = (1, \dots, 1, 0, \dots, 0)$ with d ones followed by $d(d-1)/2$ zeros, taking for instance $\mu = (2, 1, \dots, 1, 0, \dots, 0)$ would require simultaneously $t = 2$ (from the first component) and $t = 1$ (from the second), a contradiction. Hence, the class $[(\boldsymbol{P}, \mu, \Sigma)]$ admits no representative with $\mu' = 0$. □

Thus, our model of wrapped Gaussian includes wrapped Gaussians where μ is set to 0, but is also more general. We allow $\mu \neq 0$, thus we build a more general model, with more degrees of freedom.

Now that we have described the equivalence class $[\theta]$ of parameters that define the same wrapped Gaussian, it is natural to define a representative of $[\theta]$. We define it as follows:

Definition 3.26 (Representative of an equivalence class)**.** We choose as representative of the class $[\theta]$, the tuple of parameters $\theta^{\min} = (\boldsymbol{P}^{\min}, \mu^{\min}, \Sigma^{\min})$ such that $\det(\boldsymbol{P}^{\min}) = 1$. We call it the *minimal representative*.

Remark 3.27. Other minimal representatives could be chosen for the wrapped Gaussian on $\mathcal{P}_d$. Here, we choose this representative in the spirit of Remark 3.22: the minimal

representative for $\boldsymbol{P}$ is in $\mathcal{SP}_d$. Another example of minimal representative, that we chose in **de Surrel**, Lotte, et al. (2025), is to define it as the tuple of parameters that minimizes the 2-norm of μ. Then, using this representative, the wrapped Gaussian is as centered as possible in the tangent space.

One is able, given a tuple of parameters $\theta = (\boldsymbol{P}, \mu, \Sigma)$, to compute the minimal representative of the class $[\theta]$ of equivalent tuples of parameters using the following proposition:

Proposition 3.28 (Expression of the representative)**.** *Let $\theta = (\boldsymbol{P}, \mu, \Sigma) \in \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D$ be parameters. The minimal representative of the class $[\theta]$ as defined in Definition 3.26 is $\theta^{min} = (\boldsymbol{P}^{min}, \mu^{min}, \Sigma^{min})$ where*

$$\boldsymbol{P}^{min} = \det(\boldsymbol{P})^{-1/d}\boldsymbol{P}, \quad \mu^{min} = \mu + \frac{1}{d}\log(\det \boldsymbol{P})\operatorname{Vect}_{I_d}(I_d), \text{ and } \Sigma^{min} = \Sigma.$$

This minimal representative will be used when doing parameter estimation in Section 3.3.

Proof. We know, thanks to Corollary 3.24, that all the parameters in the equivalence class $[\theta]$ are of the form $(e^t\boldsymbol{P}, \mu - t\operatorname{Vect}_{I_d}(I_d), \Sigma)$, with $t \in \mathbb{R}$. Let us find $t^{\min} \in \mathbb{R}$ such that $\det(e^{t^{\min}}\boldsymbol{P}) = 1$.

$$\det(e^{t^{\min}}\boldsymbol{P}) = 1 \iff e^{dt^{\min}}\det \boldsymbol{P} = 1 \iff t^{\min} = -\frac{1}{d}\log(\det \boldsymbol{P}).$$

Thus,

$$\begin{cases} \boldsymbol{P}^{\min} = e^{t^{\min}}\boldsymbol{P} \\ \mu^{\min} = \mu - t^{\min}\operatorname{Vect}_{I_d}(I_d) \end{cases} \iff \begin{cases} \boldsymbol{P}^{\min} = \det(\boldsymbol{P})^{-1/d}\boldsymbol{P} \\ \mu^{\min} = \mu + \frac{1}{d}\log(\det \boldsymbol{P})\operatorname{Vect}_{I_d}(I_d). \end{cases} \tag{3.9}$$

□

3.2.5 Wrapped Elliptically Contoured Distribution

In this work, we focus on wrapping the multivariate Gaussian on the manifold $\mathcal{P}_d$. This multivariate Gaussian distribution belongs to the wider class of *Elliptically Contoured Distributions*. We recall the definition of an Elliptically Contoured distribution:

Definition 3.29 (Elliptically Contoured Distribution)**.** A random vector $X \in \mathbb{R}^d$ follows an Elliptically Contoured distribution if there exists $\mu \in \mathbb{R}^d$, $\Sigma \in \mathcal{P}_d$ and a function g such that X has density

$$\forall\, x \in \mathbb{R}^d,\ f_X(x) = k\det(\Sigma)^{-1/2} g\left((x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

where k is a normalizing factor. We denote $X \sim \mathrm{EC}(\mu, \Sigma, g)$.

Let us give a few examples of Euclidean elliptically contoured distributions on $\mathbb{R}^d$.

Example 3.30.
- The multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ is an elliptically contoured distribution with $g\colon t \mapsto \exp(-t/2)$.
- The multivariate t-distribution is an elliptically contoured distribution with $g\colon t \mapsto (1 + t/\nu)^{-\frac{d+\nu}{2}}$ (see V.B of chapter 1 of Delmas et al. 2024), with $\nu > 0$ the *degrees of freedom*.

One can then extend what has been done previously on the wrapped Gaussian to define *Wrapped Elliptically Contoured Distributions* just as above:

Definition 3.31 (Wrapped Elliptically Contoured Distribution)**.** Let $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D, \Sigma \in \mathcal{P}_D$ and g be a function. Then, a random vector $\boldsymbol{X}$ on $\mathcal{P}_d$ follows a Wrapped Elliptically Contoured distribution, denoted $\mathrm{WEC}(\boldsymbol{P}; \mu, \Sigma, g)$, if

$$\boldsymbol{X} = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t)), \ t \sim \mathrm{EC}(\mu, \Sigma, g).$$

One can then compute the density of $\mathrm{WEC}(\boldsymbol{P}; \mu, \Sigma, g)$ in the same way as in Theorem 3.3. This could for example lead to the definition of a multivariate t-distribution on the manifold $\mathcal{P}_d$.

3.3 Estimation of the parameters of a wrapped Gaussian

A central problem in statistics is the estimation of the parameters of a statistical model given a finite sample $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$. In this section, we tackle this estimation problem for the wrapped Gaussian defined in Section 3.2 using a *Maximum Likelihood Estimator* (MLE). After introducing and discussing the MLE, we will conduct some experiments on the manifold of SPD matrices $\mathcal{P}_d$ showing that we are able to estimate the unknown true parameters of a wrapped Gaussian.

3.3.1 The Maximum Likelihood Estimator

Let $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ be N points on the manifold $\mathcal{P}_d$ independently sampled from a wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}^\star; \mu^\star, \Sigma^\star)$ where the parameters $\theta^\star = (\boldsymbol{P}^\star, \mu^\star, \Sigma^\star)$ are unknown. We will assume in the following that the number of samples $N > D = d(d+1)/2$. Our goal is to estimate $\theta^\star$ given the sample $(\boldsymbol{X}_1, ..., \boldsymbol{X}_N)$. For this, we will use the *Maximum Likelihood Estimator* (MLE) (see Section 9.3 of Wasserman 2004). In order to define the MLE, let us first introduce the *likelihood* $\mathcal{L}_N$ of our model:

$$\forall (\boldsymbol{P}, \mu, \Sigma) \in \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D, \ \mathcal{L}_N(\boldsymbol{P}, \mu, \Sigma) = \prod_{i=1}^{N} f_{\boldsymbol{P};\mu,\Sigma}(\boldsymbol{X}_i), \tag{3.10}$$

where we recall that $f_{\boldsymbol{P};\mu,\Sigma}$ is the density of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ as given in Equation (3.2). The goal of the MLE is to maximize the likelihood $\mathcal{L}_N$. Thus, we define the MLE $\hat{\theta}_N = (\hat{\boldsymbol{P}}_N, \hat{\mu}_N, \hat{\Sigma}_N)$ as the parameters that maximize $\mathcal{L}_N$. Using the strict monotonicity of the log function, one can equivalently maximize the *log-likelihood* ℓ_N defined as $\log \mathcal{L}_N$.

Unlike the classical Euclidean multivariate Gaussian, where one has a closed-form formula for the MLE of μ and Σ, when dealing with the wrapped Gaussian, we were not able to derive a closed-form formula for the MLE $\hat{\boldsymbol{P}}$ of $\boldsymbol{P}$. We will see in the following proposition that this MLE is linked to the Fréchet mean (Fréchet 1948), for which we know that no closed-form solution exists for the manifold $\mathcal{P}_d$ endowed with the AIRM geometry (see Moakher 2005). This explains why the MLE $\hat{\boldsymbol{P}}$ of $\boldsymbol{P}$ does not have any closed-form solution.

NEW

Proposition 3.32 (MLE of $\boldsymbol{P}$ in the case of $\mathrm{WG}(\boldsymbol{P}; 0_D, I_D)$)**.** *Let us assume that we know* a priori *that* $\mu^\star = 0_D$ *and* $\Sigma^\star = I_D$*. Then, the MLE of* $\hat{\boldsymbol{P}}$ *of* $\boldsymbol{P}$ *is the solution to the following optimization problem:*

$$\hat{\boldsymbol{P}} \in \underset{\boldsymbol{P} \in \mathcal{P}_d}{\arg\min} \sum_{i=1}^{N} \left[\frac{1}{2} \delta(\boldsymbol{P}, \boldsymbol{X}_i)^2 + \log J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)) \right].$$

The solution to this problem exists.

Remark 3.33. In this case, we do not have any identifiability issue as the mean μ in the tangent space is fixed to be 0_D. Indeed, the statistical model $\{\mathrm{WG}(\boldsymbol{P}; 0_D, I_D), \boldsymbol{P} \in \mathcal{P}_d\}$ is identifiable.

Proof. Here, the MLE of $\boldsymbol{P}$ is the solution to

$$\hat{\boldsymbol{P}} \in \operatorname*{arg\,max}_{\boldsymbol{P}\in\mathcal{P}_d} \ell_N(\boldsymbol{P}) = \sum_{i=1}^{N} \log g_{0_D,I_D}(\operatorname{Vect}_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i))) - \log J_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)).$$

One can then compute

$$\begin{aligned}\log g_{0_D,I_D}(\operatorname{Vect}_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i))) &= -\frac{1}{2}\operatorname{Vect}_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i))^\top \operatorname{Vect}_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)) + C \\ &= -\frac{1}{2}\delta(\boldsymbol{P},\boldsymbol{X}_i)^2 + C\end{aligned}$$

where C is a constant independent of $\boldsymbol{P}$. The second equality uses Proposition 2.32. Moreover, using Proposition 3.5, one has that $\log J_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)) \geqslant 0$, thus this objective dominates the Fréchet functional and is therefore coercive. As the objective is continuous and coercive, a minimizer exists. This completes the proof. □

Remark 3.34. According to this proposition, the MLE $\hat{\boldsymbol{P}}$ of $\boldsymbol{P}$ is a curvature-penalized Fréchet mean. Indeed, the Fréchet mean $\bar{\boldsymbol{P}}_N$ solves the first part of the objective, however, $\log J_{\bar{\boldsymbol{P}}}(\operatorname{Log}_{\bar{\boldsymbol{P}}}(\boldsymbol{X}_i))$ is non-zero for all i almost surely. Using Proposition 3.5, one has that $\log J_{\boldsymbol{P}}(\operatorname{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)) \geqslant 0$ with equality if, and only if, there exists $t \in \mathbb{R}$ such that $\log(\boldsymbol{P}^{1/2}\boldsymbol{X}_i\boldsymbol{P}^{1/2}) = tI_d$. Thus, the penalty term is equal to zero if, and only if, all the points $\boldsymbol{X}_i$ are aligned along the geodesic $t \mapsto e^t\boldsymbol{P}$ passing through $\boldsymbol{P}$. With probability one, this is not the case, thus the penalty term is strictly positive.

In a more general setting where all three parameters $\boldsymbol{P}^\star, \mu^\star$ and $\Sigma^\star$ are unknown, we were nevertheless able to give closed-form formulas for the MLE $\hat{\mu}$ of μ and $\hat{\Sigma}$ of Σ. However, these closed-form formulas both depend on the true $\boldsymbol{P}^\star$:

Proposition 3.35 (MLE of μ and Σ)**.** *The MLE $\hat{\mu}_N$ and $\hat{\Sigma}_N$ of μ and Σ of the wrapped Gaussian are:*

$$\begin{aligned}\hat{\mu}_N &= \frac{1}{N}\sum_{i=1}^{N} \operatorname{Vect}_{\boldsymbol{P}^\star}(\operatorname{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i)), \\ \hat{\Sigma}_N &= \frac{1}{N}\sum_{i=1}^{N} \left(\operatorname{Vect}_{\boldsymbol{P}^\star}(\operatorname{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i)) - \hat{\mu}_N\right)\left(\operatorname{Vect}_{\boldsymbol{P}^\star}(\operatorname{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i)) - \hat{\mu}_N\right)^\top.\end{aligned}$$

These formulas are simply the classical MLEs of the mean and dispersion matrix of a multivariate Gaussian once the points are sent to the true tangent space $T_{\boldsymbol{P}^\star}\mathcal{P}_d$. One can note that in Galaz-Garcia et al. (2022), the authors also give a similar formula for the MLE of Σ in their Proposition 4.7, which also depends on the true $\boldsymbol{P}^\star$. Moreover, similarly to us, they were not able to derive a closed-form formula for $\boldsymbol{P}^\star$.

Proof of Proposition 3.35. Let us suppose that we know $\boldsymbol{P}^\star$. Let us compute the log-likelihood of the wrapped Gaussian model using Equation (D.2) and the density of the wrapped Gaussian given in Equation (3.2):

$$\ell_N(\mu,\Sigma) = \sum_{i=1}^{N} \log f_{\boldsymbol{P}^\star;\mu,\Sigma}(\boldsymbol{X}_i) = \sum_{i=1}^{N} \log g_{\mu,\Sigma}(\operatorname{Vect}_{\boldsymbol{P}^\star}(\operatorname{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i))) - \log J_{\boldsymbol{P}^\star}(\operatorname{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i)).$$

One can see that only the first term depends on μ and Σ. Thus, by denoting $t_i = \operatorname{Vect}_{\boldsymbol{P}^\star}(\operatorname{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i)) \in \mathbb{R}^D$, maximizing the above log-likelihood amounts to maximizing

$$\sum_{i=1}^{N} \log g_{\mu,\Sigma}(t_i),$$

over μ and Σ. Thus, computing the MLE of μ and Σ can be done in the exact same way as in the Euclidean case, leading to the same estimators where $t_i = \mathrm{Vect}_{\boldsymbol{P}^\star}(\mathrm{Log}_{\boldsymbol{P}^\star}(\boldsymbol{X}_i))$. □

NEW

Finally, we are able to simplify the optimization objective of the MLE $\hat{\boldsymbol{P}}$ of $\boldsymbol{P}$ in the general case:

Proposition 3.36 (MLE of $\boldsymbol{P}$ in the general setting)**.** *In the general case where all the parameters of the wrapped Gaussian are unknown, the MLE $\hat{\boldsymbol{P}}_N$ of $\boldsymbol{P}$ is such that $(\hat{\boldsymbol{P}}_N, \hat{\mu}(\hat{\boldsymbol{P}}_N), \hat{\Sigma}(\hat{\boldsymbol{P}}_N))$ maximizes the likelihood $\mathcal{L}_N$ if and only if*

$$\hat{\boldsymbol{P}}_N \in \operatorname*{arg\,min}_{\boldsymbol{P}\in\mathcal{P}_d} \mathcal{F}_N(\boldsymbol{P}), \qquad \mathcal{F}_N(\boldsymbol{P}) := \frac{N}{2}\log\det\hat{\Sigma}(\boldsymbol{P}) + \sum_{i=1}^{N}\log J_{\boldsymbol{P}}\big(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)\big),$$

where $\hat{\mu}(\boldsymbol{P})$ and $\hat{\Sigma}(\boldsymbol{P})$ are the estimators of Proposition 3.35 evaluated at $\boldsymbol{P}$. Moreover, $\mathcal{F}_N$ is invariant along the flat direction, i.e. $\mathcal{F}_N(e^t\boldsymbol{P}) = \mathcal{F}_N(\boldsymbol{P})$ for all $t \in \mathbb{R}$, so that the above arg min *is a union of geodesics $\{e^t\boldsymbol{P}\}_{t\in\mathbb{R}}$, in accordance with Theorem 3.18; a unique representative is then obtained through Proposition 3.28.*

Remark 3.37. Because of the non-identifiability of the full wrapped Gaussian model $\{\mathrm{WG}(\boldsymbol{P};\mu,\Sigma), \boldsymbol{P}\in\mathcal{P}_d, \mu\in\mathbb{R}^D \text{ and } \Sigma\in\mathcal{P}_D\}$ (see Theorem 3.18), we know that this optimization problem does not have a unique solution.

Proof. Since $\ell_N = \log\mathcal{L}_N$ and the supremum of a function of several variables may be computed iteratively,

$$\sup_{\boldsymbol{P},\mu,\Sigma}\ell_N(\boldsymbol{P},\mu,\Sigma) = \sup_{\boldsymbol{P}\in\mathcal{P}_d}\Big(\sup_{\mu\in\mathbb{R}^D,\,\Sigma\in\mathcal{P}_D}\ell_N(\boldsymbol{P},\mu,\Sigma)\Big),$$

and a triplet $(\boldsymbol{P},\mu,\Sigma)$ attains the left-hand side if and only if $\boldsymbol{P}$ attains the outer supremum on the right and (μ,Σ) attains the inner one. As $N > D$, the points $t_i := \mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i))$ are almost surely in general position, so $\hat{\Sigma}(\boldsymbol{P})\in\mathcal{P}_D$ and, by Proposition 3.35 applied at the point $\boldsymbol{P}$, the inner supremum is attained exactly at $(\hat{\mu}(\boldsymbol{P}),\hat{\Sigma}(\boldsymbol{P}))$.

It remains to compute the profiled log-likelihood $\ell_N^{\mathrm{prof}}(\boldsymbol{P}) := \ell_N(\boldsymbol{P},\hat{\mu}(\boldsymbol{P}),\hat{\Sigma}(\boldsymbol{P}))$. Writing $\hat{\mu}=\hat{\mu}(\boldsymbol{P})$ and $\hat{\Sigma}=\hat{\Sigma}(\boldsymbol{P})$ for brevity, each summand of the quadratic form is a scalar, hence equal to its trace, and

$$\begin{aligned}
\sum_{i=1}^{N}\log g_{\hat{\mu},\hat{\Sigma}}(t_i) &= -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det\hat{\Sigma} - \frac{1}{2}\sum_{i=1}^{N}(t_i-\hat{\mu})^\top\hat{\Sigma}^{-1}(t_i-\hat{\mu})\\
&= -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det\hat{\Sigma} - \frac{1}{2}\sum_{i=1}^{N}\mathrm{tr}\left(\hat{\Sigma}^{-1}(t_i-\hat{\mu})(t_i-\hat{\mu})^\top\right)\\
&= -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det\hat{\Sigma} - \frac{1}{2}\mathrm{tr}\Big(\hat{\Sigma}^{-1}\underbrace{\sum_{i=1}^{N}(t_i-\hat{\mu})(t_i-\hat{\mu})^\top}_{=N\hat{\Sigma}}\Big)\\
&= -\frac{ND}{2}\log(2\pi) - \frac{N}{2}\log\det\hat{\Sigma} - \frac{ND}{2},
\end{aligned}$$

where the last equality uses $\mathrm{tr}(\hat{\Sigma}^{-1}\hat{\Sigma}) = \mathrm{tr}(I_D) = D$. Since $J_{\boldsymbol{P}} \geqslant 1 > 0$ by Proposition 3.5, Theorem 3.3 gives

$$\ell_N^{\mathrm{prof}}(\boldsymbol{P}) = -\mathcal{F}_N(\boldsymbol{P}) - \frac{ND}{2}(1+\log(2\pi)),$$

and the additive constant does not depend on $\boldsymbol{P}$, which proves the stated equivalence.

Finally, let $t\in\mathbb{R}$. One has $\mathrm{Log}_{e^t\boldsymbol{P}}(\boldsymbol{X}_i) = e^t\boldsymbol{P}^{1/2}(\boldsymbol{S}_i - tI_d)\boldsymbol{P}^{1/2}$ with $\boldsymbol{S}_i = \log(\boldsymbol{P}^{-1/2}\boldsymbol{X}_i\boldsymbol{P}^{-1/2})$, so that $t_i(e^t\boldsymbol{P}) = t_i(\boldsymbol{P}) - t\ \mathrm{Vect}_{I_d}(I_d)$. The centered quantities $t_i - \hat{\mu}$ are therefore unchanged,

hence so is $\hat{\Sigma}$; and the eigenvalues of $\boldsymbol{S}_i$ are all shifted by $-t$, leaving their pairwise differences, and hence $J_{\boldsymbol{P}}$ by Proposition 3.4, unchanged. Thus, $\mathcal{F}_N(e^t\boldsymbol{P}) = \mathcal{F}_N(\boldsymbol{P})$. □

In practice, to compute the MLE, we used a Riemannian optimization algorithm (Boumal 2023) on the product manifold of parameters $\Theta = \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D$. However, thanks to Proposition 3.36, we only need to solve this optimization problem over $\boldsymbol{P}$. At each iteration of the Riemannian optimization algorithm, the corresponding optimal values of μ and Σ are computed explicitly. This significantly improved the convergence of the algorithm. We implemented this MLE procedure in Python using the Riemannian optimization toolbox Pymanopt (Townsend et al. 2016), together with the automatic differentiation capabilities provided by JAX (Bradbury et al. 2018). This implementation will be used to perform the numerical experiments in Section 3.3.3.

3.3.2 Method of moments

Another estimation method is the method of moments (see Section 9.2 of Wasserman 2004) based on the computation of the moments of a distribution. This method has been used for wrapped distributions in E. Chevallier and Guigui (2020) and E. Chevallier, D. Li, et al. (2022) to estimate the parameters. In our case, one can use the method of moments when one knows *a priori* that $\mu^\star = 0_D$. Then, using Proposition 3.10, one knows that a mean of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}^\star; 0_D, \Sigma^\star)$ is $\boldsymbol{P}^\star$ that can be estimated using the Fréchet mean (Fréchet 1948):

$$\mathfrak{G}(\boldsymbol{X}_1, ..., \boldsymbol{X}_N) = \underset{\boldsymbol{P}\in\mathcal{P}_d}{\arg\min} \sum_{i=1}^{N} \delta(\boldsymbol{X}_i, \boldsymbol{P})^2.$$

As mentioned in Remark 2.24, this Fréchet mean has no closed-form solution but can be numerically estimated using a Riemannian gradient descent. Finally, using Proposition 3.35, one can estimate $\Sigma^\star$ by sending the points $(\boldsymbol{X}_1, ..., \boldsymbol{X}_N)$ to the tangent space $T_{\mathfrak{G}(\boldsymbol{X}_1,...,\boldsymbol{X}_N)}\mathcal{P}_d$ and computing the classical Euclidean MLE for the dispersion matrix.

However, when $\mu^\star \neq 0_D$, using the Fréchet mean and then applying Proposition 3.35 will not produce a correct estimation of the true parameters $(\boldsymbol{P}^\star, \mu^\star, \Sigma^\star)$. Indeed, suppose that we estimate $\boldsymbol{P}^\star$ using the Fréchet mean $\mathfrak{G}(\boldsymbol{X}_1, ..., \boldsymbol{X}_N)$. Then, computing the Riemannian gradient of the Fréchet mean $\mathfrak{G}(\boldsymbol{X}_1, ..., \boldsymbol{X}_N)$ and using a first order optimality condition (Propositions 4.5 and 4.6 of Boumal 2023), gives us that

$$\sum_{i=1}^{N} \mathrm{Log}_{\mathfrak{G}(\boldsymbol{X}_1,...,\boldsymbol{X}_N)}(\boldsymbol{X}_i) = 0.$$

Thus, applying Proposition 3.35 to compute the MLE $\hat{\mu}_N$ of $\mu^\star$, and using the linearity of the vectorization $\mathrm{Vect}_{\mathfrak{G}(\boldsymbol{X}_1,...,\boldsymbol{X}_N)}$, one can see that $\hat{\mu}_N = 0_D$ which is not a good estimator for $\mu^\star \neq 0_D$. Therefore, we will not use the Fréchet mean to estimate $\boldsymbol{P}$ in a general setting.

3.3.3 Numerical experiments

We conducted some synthetic experiments to evaluate the performance of the MLE. More precisely, we want to answer the following question: is the MLE given in Proposition 3.36 able to correctly estimate the parameters of the wrapped Gaussian, and does it scale well? We will investigate both the scaling with respect to the number of samples N and the dimension d. For this, we generated a synthetic dataset by sampling N points from a wrapped Gaussian in $\mathcal{P}_d$ whose minimal parameters $(\boldsymbol{P}^\star, \mu^\star, \Sigma^\star)$ are known. Here, $\Sigma^\star$ is always chosen to be a full (not diagonal) SPD matrix. $\boldsymbol{P}^\star$ and $\Sigma^\star$ were randomly sampled by sampling eigenvalues uniformly

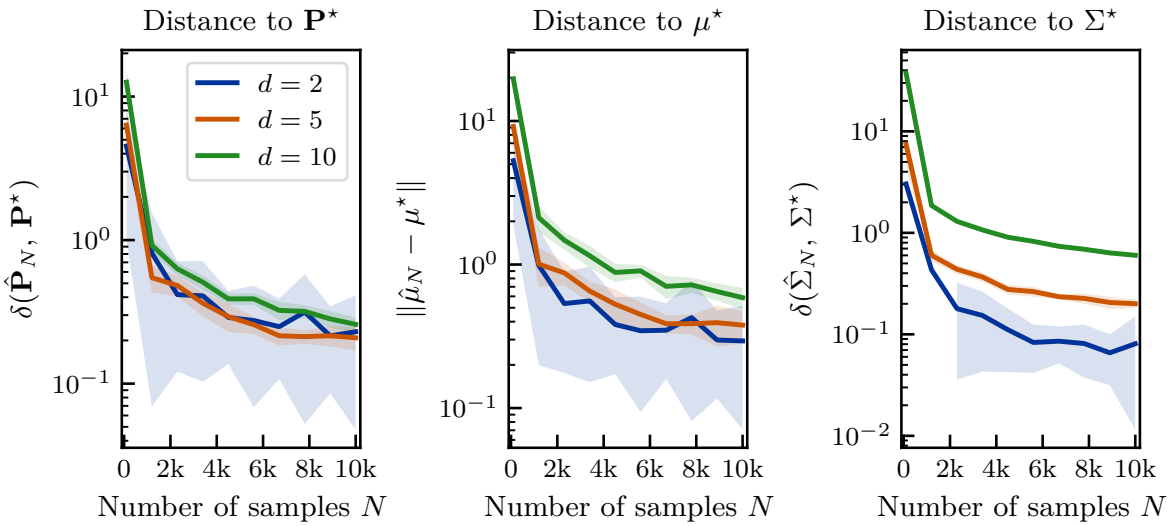


Figure 3.5. Results of the synthetic experiment on the estimation of parameters of a wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}^\star; \mu^\star, \Sigma^\star)$ using an MLE. Each panel corresponds to the estimation error of one parameter (left: $\boldsymbol{P}$, middle: μ and right: Σ). The different colors correspond to different dimensions of the manifold $\mathcal{P}_d$.

in $[0.5, 2.0]$ and by sampling the orthonormal eigenvectors uniformly[3]. $\mu^\star$ was sampled from a $\mathcal{N}(0, I_D)$. Then, we optimized the MLE to find $(\hat{\boldsymbol{P}}_N, \hat{\mu}_N, \hat{\Sigma}_N)$, and compared the true and minimal estimated parameters: for $\boldsymbol{P}$ and Σ, we compared them using the AIRM distance and for μ, we used $\|\cdot\|_2$. In this experiment, we looked at how the estimation error evolves when the number of samples N grows from 100 to 10,000. We also compared different dimensions $d \in \{2, 5, 10\}$, and we repeated the experiment 10 times with different true parameters $\theta^\star$. The results of this experiment can be found in Figure 3.5. We can see that, as one would expect, as the number N of points sampled grows, the estimation error decreases. However, the estimation error does not seem to converge to 0, which would indicate that the estimators may be biased (at least when the dispersion matrix Σ is full and not diagonal).

Moreover, we can remark that the dimension d does not affect the results of the estimation of $\boldsymbol{P}$, it slightly affects the estimation of μ, and it strongly affects the estimation of Σ. The higher the dimension d, the higher the error $\delta(\hat{\Sigma}_N, \Sigma^\star)$. This is consistent, as the number of parameters of Σ grows as $O(d^4)$, so even a small increase of the dimension d leads to a significant increase in the estimation error of Σ. In order to reduce the number of parameters to estimate, one can impose *a priori* the dispersion matrix Σ to be diagonal, reducing the number of parameters to estimate to only $O(d^2)$. We conducted some experiments in this case, with a setup similar to the one above but taking only the diagonal of the true $\Sigma^\star$ and restricting ourselves to dimension $d = 10$. We give the results in Figure 3.6. One can see that when $\hat{\Sigma}_N$ is only diagonal, the estimation error decreases significantly for the same number of samples N. This is consistent, as the number of coefficients to estimate is reduced when Σ is diagonal. Therefore, this setting of Σ diagonal can be a good choice when the number of samples is limited.

3.4 Extension to other manifolds

In what precedes, we have defined the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ exclusively on the manifold $\mathcal{P}_d$ of SPD matrices. However, one can see thin Definition D.1 does not depend

[3]This is done by sampling $\boldsymbol{X} \in \mathbb{R}^{d \times d}$ whose entries follow a $\mathcal{N}(0, 1)$ distribution, computing its QR decomposition and using the resulting Q as the matrix of the eigenvectors. This is equivalent to sampling from the Haar measure on $O(d)$. See Proposition 7.3 of Eaton (2007) for a proof.

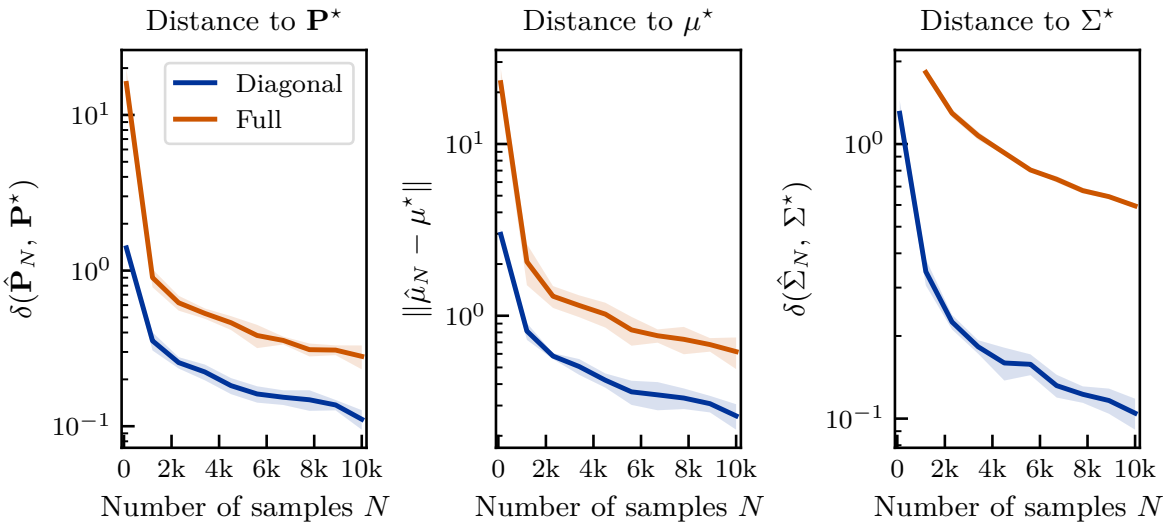


Figure 3.6. Comparison of the estimation of the parameters of a wrapped Gaussian when the dispersion matrix Σ is diagonal or full. The dimension of the SPD matrices in this experiment is $d = 10$.

on this specific choice of the manifold $\mathcal{P}_d$, so one could wonder if it would be possible to extend this definition to more general spaces. An important property of the manifold $\mathcal{P}_d$ (and one mostly hidden in the previous proofs) is that the exponential mapping $\mathrm{Exp}_{\boldsymbol{P}}$ is a global diffeomorphism from $T_{\boldsymbol{P}}\mathcal{P}_d$ to $\mathcal{P}_d$. As the Euclidean Gaussian has an infinite support, the exponential mapping $\mathrm{Exp}_{\boldsymbol{P}}$ will generate an infinitely supported distribution on the manifold $\mathcal{P}_d$ that does not "wrap" back on itself. This is for example what happens when wrapping a distribution on the circle, leading to a complex density (see Equation (2.12)). A particular class of manifolds for which the exponential mapping is a global diffeomorphism is that of *Hadamard manifolds*, and a wrapped Gaussian can be defined on them as previously. Let us give more details on this.

3.4.1 Wrapped Gaussian on a Hadamard manifold

Let us start by giving some definitions and important properties of Hadamard manifolds. We refer to Shiga (1984) for more technical details on Hadamard manifolds.

Definition 3.38 (Hadamard manifold)**.** A Hadamard manifold is a complete, simply connected Riemannian manifold of non-positive sectional curvature everywhere.

Example 3.39. Examples of Hadamard manifolds are the Euclidean space $\mathbb{R}^d$ (of zero curvature), the hyperbolic space $\mathbb{H}^d$ (of constant curvature of -1) and the manifold of Symmetric Positive Definite matrices $\mathcal{P}_d$ (of curvature between $-1/2$ and 0 when equipped with the affine-invariant metric, see Proposition 2.21), or the manifold of Hermitian positive definite matrices (Sra and R. Hosseini 2015). Other Hadamard manifolds can be built by taking the product of Hadamard manifolds.

In the following, we will consider a generic Hadamard manifold of dimension d, denoted by $\mathcal{M}$. As, by definition, $\mathcal{M}$ is a complete connected Riemannian manifold, each pair of points $p, q \in \mathcal{M}$ can be joined by a unique minimizing geodesic (see Corollary 6.21 of J. M. Lee 2018). Hadamard manifolds are a good setup for our study thanks to the Cartan-Hadamard theorem (Theorem 12.8 of J. M. Lee 2018) which states the following:

Theorem 3.40 (Cartan-Hadamard's theorem)**.** *Let $\mathcal{M}$ be a Hadamard manifold. For $p \in \mathcal{M}$, the exponential map $\mathrm{Exp}_p \colon T_p\mathcal{M} \to \mathcal{M}$ is a global diffeomorphism from $T_p\mathcal{M}$ onto $\mathcal{M}$.*

This theorem ensures the global definition of the exponential map Exp_p and of the logarithmic map $\mathrm{Log}_p \colon \mathcal{M} \to T_p\mathcal{M}$ (defined as the inverse of Exp_p) on $\mathcal{M}$. One is able to vectorize the tangent space of a Hadamard manifold $\mathcal{M}$ by defining a linear isometry between $T_p\mathcal{M}$ (where $p \in \mathcal{M}$) and $\mathbb{R}^d$ in the same way as was done for $\mathcal{P}_d$ in Definition 2.31. This requires the choice of a reference point o in the space $\mathcal{M}$. For example, when $\mathcal{M} = \mathcal{P}_d$, the natural choice is $o = I_d$, for $\mathcal{M} = \mathbb{R}^d$, $o = 0_d$ and for[4] $\mathcal{M} = \mathbb{H}^d$, $o = (1, 0, \ldots, 0) \in \mathbb{R}^{d+1}$.

Using all of this, we are able to define the wrapped Gaussian $\mathrm{WG}(p; \mu, \Sigma)$ on a Hadamard manifold $\mathcal{M}$ in exactly the same way as in Definition D.1, or using Equation (D.1):

$$\mathrm{WG}(p; \mu, \Sigma) = \left(\mathrm{Exp}_p \circ \mathrm{Vect}_p^{-1}\right) \# \mathcal{N}(\mu, \Sigma). \tag{3.11}$$

Most of the work that has been done on the manifold $\mathcal{P}_d$ of SPD matrices is still valid on a general Hadamard manifold. Let us be more precise. The density of the wrapped Gaussian will have the same expression as the one given in Theorem 3.3. However, in the general case, one does not have a closed form for the Jacobian determinant J_p of the exponential mapping. This Jacobian determinant is linked to the curvature of the manifold $\mathcal{M}$. The authors of E. Chevallier, D. Li, et al. (2022) give a formula for $J_p(\cdot)$ when the manifold is affine and locally symmetric. However, it relies on the eigenvalues of a linear map defined using the curvature tensor (see Theorem 4.1 of E. Chevallier, D. Li, et al. 2022), which are not trivial to compute. For the rescaling property (Proposition 3.8), point 2 and point 3 are still valid, but the first point needs to be rewritten in a more general form: let $X \sim \mathrm{WG}(p; \mu, \Sigma)$ and let $q \in \mathcal{M}$. One has that

$$(\mathrm{Exp}_q \circ \Gamma_{o \to q} \circ \Gamma_{p \to o} \circ \mathrm{Log}_p)(X) \sim \mathrm{WG}(q; \mu, \Sigma),$$

where $\Gamma_{p \to o}$ (resp. $\Gamma_{o \to q}$) is the parallel transport from p to o (resp. from o to q) along the unique geodesic connecting p to o (resp. o to q). The known moments of the wrapped Gaussian given in Proposition 3.10 and Proposition 3.11 stay valid on a general Hadamard manifold $\mathcal{M}$. For the wrapped Central Limit Theorem (CLT) given in Theorem 3.16, a similar line of reasoning can lead to a general wrapped CLT; however, Equation (3.4) has this simple expression thanks to specific properties of SPD matrices. Thus, one cannot expect to have a simple expression for the wrapped CLT on a general Hadamard manifold.

Now comes an important question: what about the identifiability of the wrapped Gaussian defined on a general Hadamard manifold $\mathcal{M}$? We have already seen in Section 3.2.4 that the wrapped Gaussian defined on the manifold of SPD matrices $\mathcal{P}_d$ is not identifiable (see Theorem 3.18). But does this result generalize to an arbitrary Hadamard manifold $\mathcal{M}$? While we do not have a complete answer for the general case, let us study two specific examples: the Euclidean space $\mathbb{R}^d$ and the hyperbolic space $\mathbb{H}^d$.

The Euclidean space $\mathbb{R}^d$ Endowed with its classical inner product $\langle \cdot, \cdot \rangle_2$, the Euclidean space $\mathbb{R}^d$ is of zero curvature and thus is a Hadamard manifold. One can thus define the wrapped Gaussian on $(\mathbb{R}^d, \langle \cdot, \cdot \rangle_2)$ which boils down to

$$\mathrm{WG}(p; \mu, \Sigma) = \mathcal{N}(\mu, \Sigma) + p = \mathcal{N}(\mu + p, \Sigma).$$

Thus, the wrapped Gaussian in this setting is the classical multivariate Gaussian on $\mathbb{R}^d$. However, one can see that, on $\mathbb{R}^d$, the two parameters p and μ compete with each other. Therefore, the wrapped Gaussian on $\mathbb{R}^d$ is not identifiable, as one has

$$\mathrm{WG}(p_1; \mu_1, \Sigma_1) = \mathrm{WG}(p_2; \mu_2, \Sigma_2) \iff p_1 + \mu_1 = p_2 + \mu_2 \text{ and } \Sigma_1 = \Sigma_2, \tag{3.12}$$

[4] For the Lorentz model of the hyperbolic space, see Benedetti and Petronio (1992).

which does not imply $p_1 = p_2$ and $\mu_1 = \mu_2$. Thus, for the Euclidean space $\mathbb{R}^d$, imposing $\mu = 0$ in the definition of a wrapped Gaussian does not reduce its expressiveness and resolves the identifiability issue (the wrapped Gaussian is then simply the multivariate Gaussian on $\mathbb{R}^d$, which is identifiable).

Remark 3.41. As the curvature is preserved by pullback operators, pullback Euclidean manifolds are Hadamard manifolds (see Proposition 13.9 of J. M. Lee 2012, for the definition of a pullback Euclidean manifold). This simple flat geometry is used in several works, such as the Log-Euclidean geometry on SPD matrices (Arsigny et al. 2006b), or to build other geometries such as in Z. Chen, Yue Song, T. Xu, et al. (2024), Augenstein et al. (2024), Collas, Ju, et al. (2026), Diepeveen et al. (2025), and González-Duque et al. (2025). One can define the wrapped Gaussian on such a manifold in the same way as done above.

The hyperbolic space Hyperbolic spaces are Riemannian manifolds of constant negative curvature. There are four main equivalent models used for the hyperbolic geometry: the Klein model, the Poincaré ball, the Poincaré half-plane model and the Lorentz (hyperboloid/Minkowski) model (see Benedetti and Petronio (1992) or chapter 3 of J. M. Lee (2018) for more details on these models). All of these models are mutually isometric, so one can choose the most convenient one. Moreover, the definition of the wrapped Gaussian does not depend on the model. An important fact about the hyperbolic space when defining a wrapped Gaussian is that we know how to compute explicitly the Jacobian determinant of the exponential mapping Exp_p. We refer to Section 3.2 of Nagano et al. (2019) for the closed-form expression of this Jacobian determinant.

After defining the wrapped Gaussian on the hyperbolic space $\mathbb{H}^d$, we have an important result on the identifiability of the wrapped Gaussian on $\mathbb{H}^d$.

Theorem 3.42 (Identifiability of the wrapped Gaussian on the hyperbolic space)**.** *The wrapped Gaussian* ***is identifiable*** *on the hyperbolic space, meaning that*

$$\mathrm{WG}(p; \mu, \Sigma) = \mathrm{WG}(q; \nu, \Gamma) \iff p = q, \mu = \nu \text{ and } \Sigma = \Gamma.$$

This positive result (the wrapped Gaussian is identifiable on $\mathbb{H}^d$) stands in contrast to the negative results we have on $\mathcal{P}_d$ (see Theorem 3.18) and $\mathbb{R}^d$ (see Equation (3.12)). Thus, the identifiability of the wrapped Gaussian depends on the space on which it is defined. The proof of this statement is given in Appendix A.2.2, and we simply give some ideas of the proof here.

Main ideas of the proof. For this proof, we consider the Poincaré ball model for the hyperbolic space $\mathbb{H}^d$. We refer to Mathieu et al. (2019) for a detailed presentation of this model[5]. The sufficient condition ($\Leftarrow$) is trivial. For the necessary one ($\Rightarrow$), we focus on showing the result for $\mathrm{WG}(0_d; \mu, \Sigma)$. We suppose that $\mathrm{WG}(0_d; \mu, \Sigma) = \mathrm{WG}(q; \nu, \Gamma)$. This is equivalent to an equality of the densities. The main idea is to study the limits of both terms of this equality as the tangent vector $u \in T_{0_d}\mathbb{H}^d$ satisfies $\|u\| \to \infty$. We are able to show that they do not have the same limit if $q \neq 0_d$. This implies $q = 0_d$. Then, we conclude by using the identifiability of the multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ to show that $\nu = \mu$ and $\Gamma = \Sigma$. □

One can now ask about the difference between the spaces $\mathbb{R}^d$ and $\mathcal{P}_d$ and the hyperbolic space $\mathbb{H}^d$ and how the structure of the manifold $\mathcal{M}$ is related to the identifiability of the wrapped Gaussian. Let us recall a theorem from differential geometry, the *de Rham decomposition* originally proven by de Rham (1952) and that can be found in Theorem 6.11 of Sakai (1996). This decomposition gives us a canonical splitting of a Hadamard manifold.

NEW

[5] As all models of the hyperbolic space are isometric, proving this result in a specific model does not affect the general result.

Theorem 3.43 (de Rham decomposition)**.** *Let $\mathcal{M}$ be a complete simply connected Riemannian manifold of dimension d. Then there is an isometry, unique up to permutation of factors,*

$$\mathcal{M} \cong \mathbb{R}^k \times \mathcal{M}_1 \times \cdots \times \mathcal{M}_m, \tag{3.13}$$

where $\mathbb{R}^k$ is the Euclidean space with the canonical Riemannian metric (possibly reduced to a point if $k = 0$), and $\mathcal{M}_i, (i = 1, ..., m)$ are complete simply connected irreducible Riemannian manifolds.

In this theorem, $\mathbb{R}^k$ is the maximal flat factor of the space $\mathcal{M}$. Let us look at the de Rham decompositions of the spaces $\mathbb{R}^d$, $\mathcal{P}_d$ and $\mathbb{H}^d$. For $\mathbb{R}^d$, it is immediate that the de Rham decomposition is $\mathbb{R}^d$. For $\mathcal{P}_d$, thanks to Remark 3.22, one has that $\mathcal{P}_d \cong \mathbb{R} \times \mathcal{SP}_d$, where $\mathcal{SP}_d$ is the set of SPD matrices of determinant 1. Moreover, one has that $\mathcal{SP}_d \cong \mathrm{SL}(d)/\mathrm{SO}(d)$, where $\mathrm{SL}(d)$ is the set of invertible matrices of determinant 1 and $\mathrm{SO}(d)$ the set of orthogonal matrices of determinant 1. This is related to the quotient structure of $\mathcal{P}_d$ given in Proposition 2.9. According to Table V of Chapter X of Helgason (1979), the space $\mathrm{SL}(d)/\mathrm{SO}(d)$ is an irreducible symmetric space, thus the de Rham decomposition of $\mathcal{P}_d$ is

$$\mathcal{P}_d \cong \mathbb{R} \times \mathrm{SL}(d)/\mathrm{SO}(d). \tag{3.14}$$

Finally, the hyperbolic space $\mathbb{H}^d$ is irreducible so it is its own de Rham decomposition.

We summarize these three cases in Table 3.2.

Space	de Rham decomposition	Flat factor dim.	Identifiable?
$\mathbb{R}^d$	$\mathbb{R}^d$	$k = d$	✗
$\mathcal{P}_d$	$\mathbb{R} \times \mathrm{SL}(d)/\mathrm{SO}(d)$	$k = 1$	✗
$\mathbb{H}^d$	$\mathbb{H}^d$	$k = 0$	✓

Table 3.2. De Rham decompositions of three Hadamard manifolds and identifiability of the wrapped Gaussian on each. The flat factor dimension k appears to be the key quantity.

Thanks to these insights, we can formulate a first proposition:

Proposition 3.44. *If, in the de Rham decomposition of $\mathcal{M}$, the flat factor is **not** reduced to a point, i.e. $k > 0$, then the wrapped Gaussian is **not** identifiable.*

Proof. We give here the main ideas of the proof. Let us consider the de Rham decomposition of $\mathcal{M}$ written as follows, using the notations of Theorem 3.43:

$$\mathcal{M} = \mathbb{R}^k \times \mathcal{N},$$

where $\mathcal{N} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_m$. Let us suppose that $k > 0$, thus the flat factor $\mathbb{R}^k$ is not reduced to a single point. Then, when Σ is block-diagonal with respect to the above splitting, the wrapped Gaussian $\mathrm{WG}_{\mathcal{M}}(p; \mu, \Sigma)$ defined on $\mathcal{M}$ can be written as:

$$\mathrm{WG}_{\mathcal{M}}(p; \mu, \Sigma) = \mathrm{WG}_{\mathbb{R}^k}(p_{\mathbb{R}^k}; \mu_{\mathbb{R}^k}, \Sigma_{\mathbb{R}^k}) \otimes \mathrm{WG}_{\mathcal{N}}(p_{\mathcal{N}}; \mu_{\mathcal{N}}, \Sigma_{\mathcal{N}}),$$

where $\mathrm{WG}_{\mathbb{R}^k}(p_{\mathbb{R}^k}; \mu_{\mathbb{R}^k}, \Sigma_{\mathbb{R}^k})$ is a wrapped Gaussian on $\mathbb{R}^k$ and $\mathrm{WG}_{\mathcal{N}}(p_{\mathcal{N}}; \mu_{\mathcal{N}}, \Sigma_{\mathcal{N}})$ is defined on $\mathcal{N}$.

Using the non-identifiability of the wrapped Gaussian on $\mathbb{R}^k$ as given in Equation (3.12), we have that $\mathrm{WG}_{\mathbb{R}^k}(p_{\mathbb{R}^k}; \mu_{\mathbb{R}^k}, \Sigma_{\mathbb{R}^k})$ is not identifiable, and thus, $\mathrm{WG}_{\mathcal{M}}(p; \mu, \Sigma)$ is not identifiable either. □

When the flat factor in the de Rham decomposition is reduced to a point ($k = 0$), as is the case for $\mathrm{SL}(d)/\mathrm{SO}(d)$ or for $\mathbb{H}^d$, we conjecture that the wrapped Gaussian is identifiable. Thus, we formulate the following conjecture on the identifiability of the wrapped Gaussian on a Hadamard manifold $\mathcal{M}$.

Conjecture 3.45. *Let $\mathcal{M} \cong \mathbb{R}^k \times \mathcal{M}_1 \times \cdots \times \mathcal{M}_m$ be a Hadamard manifold, with $\mathbb{R}^k$ its maximal flat factor. Then, the wrapped Gaussian is identifiable on $\mathcal{M}$ if and only if $\mathcal{M}$ has no flat factor, i.e. $k = 0$.*

According to this conjecture, the redundancy in the parameters of the wrapped Gaussian lies only in the flat factor of the space. On $\mathbb{R}^d$, the flat factor is everything, thus in all directions of $\mathbb{R}^d$, the wrapped Gaussian has some redundancy. On $\mathcal{P}_d$, the flat factor is only of dimension 1 as given by Equation (3.14), and one can see, using Theorem 3.18, that the redundancy of the wrapped Gaussian on $\mathcal{P}_d$ is only "on a line", a one-dimensional space. Conversely, on $\mathbb{H}^d$, where there is no flat factor, there is no redundancy at all.

3.5 Conclusion

This chapter was devoted to the introduction and the study of the wrapped Gaussian distribution on the manifold $\mathcal{P}_d$ of SPD matrices. We started by defining this wrapped Gaussian, studying some of its properties (density, moments, rescaling, wrapped CLT). Then, a major focus was put on the identifiability issue we were faced with. Indeed, by building a very general model for which we allowed the mean μ of the Euclidean distribution in the tangent space to be different from 0, we encountered a conflict between the anchor $\boldsymbol{P}$ of the wrapped Gaussian and the mean μ in the tangent space. These two parameters compete with each other, and we fully characterized the relationship between the two in Theorem 3.18. This helped us solve the identifiability issue by defining an equivalence relation and a minimal representation of the orbits. We then tackled the essential estimation problem, and showed that, given enough samples from an unknown wrapped Gaussian, we were able to estimate its parameters using an MLE. Finally, we started to extend this wrapped Gaussian to other spaces, and proposed a conjecture on the general identifiability of the wrapped Gaussian.

Looking back at the introduction of this chapter, and the list of desired properties we wanted for our distribution, we can see in Table 3.1 that the wrapped Gaussian we introduced fulfills all the properties. Moreover, the proposed wrapped Gaussian has more degrees of freedom than the two other Gaussian distributions: compared to the isotropic Gaussian, the main addition is the *anisotropy* of the wrapped Gaussian thanks to its full dispersion matrix. This anisotropy property proves crucial in real-life applications, such as distinguishing anesthetized patients from awake ones, as we will show in Chapter 6. Compared to the entropy-maximizing Gaussian, the wrapped Gaussian is also a larger model (thanks to the mean μ in the tangent space), but is above all more usable in practice (sampling, closed-form density, roles of the parameters).

This wrapped Gaussian gives us an answer to our main question 1: indeed, this statistical model is geometry-aware, as it takes into account the geometry of the manifold on which it is defined. This statistical model will be the main tool that we will use in the following. Indeed, the Gaussian distribution is a very common prior when doing machine learning, and a lot of algorithms are based on this distribution. Now that we have a handy Gaussian on the manifold of SPD matrices, we can see how these learning methods could be transferred to $\mathcal{P}_d$. In the next chapter, we will see that a lot of the learning tools that already exist on $\mathcal{P}_d$ rely on the isotropic Gaussian or on the wrapped Gaussian. For those that rely only on the isotropic Gaussian, we will see how we can generalize them to anisotropic tools using the wrapped Gaussian. These new anisotropic and wrapped algorithms will be tested on real-life data, first in Chapter 4 and then in a deeper study, in Chapter 6. Finally, we will also use the wrapped Gaussian to build new *Flow Matching* models on $\mathcal{P}_d$ in Chapter 5. An interesting extension, whose contours have begun to be outlined in Section 3.2.5, is the class of *Wrapped Elliptically Contoured Distributions*. If these distributions inherit their Euclidean advantages while having properties similar to those of the wrapped Gaussian, they could help design

more robust distributions on $\mathcal{P}_d$.

We must still mention that, in order to get these benefits from the wrapped Gaussian, we relied on the tangent space, a local linear approximation of the space. Thus, the wrapped Gaussian, as mentioned in Section 2.3.6, is an *extrinsic method*. We will indeed see in Chapter 4 that most of the extrinsic methods from Table 2.4 can be rewritten in terms of the wrapped Gaussian. However, another major advantage of the wrapped Gaussian in this case is that the parameter controlling the tangent space, $\boldsymbol{P}$, is a real parameter of our statistical model, not an *ex nihilo* parameter added to Euclidean methods. This will improve the flexibility of the extrinsic methods.

4

Chapter 4

Building a probabilistic framework for ML on $\mathcal{P}_d$

Contents

4.1 Introduction . . . 93
4.2 The two Gaussians of interest . . . 95
4.2.1 The isotropic Gaussian . . . 95
4.2.2 The wrapped Gaussian . . . 96
4.3 Classification under the Gaussian lens . . . 96
4.3.1 Minimum Distance to Mean (MDM) . . . 97
4.3.2 Tangent Space LDA/QDA . . . 98
4.3.3 Wrapped Discriminant Analysis . . . 99
4.3.4 Other probabilistic classifiers for SPD matrices . . . 101
4.3.5 Numerical experiments comparing the classifiers . . . 101
4.4 Outlier detection *à la* Gaussian . . . 104
4.5 Kernels with a Gaussian flavor . . . 106
4.5.1 The AIRM Kernel . . . 106
4.5.2 The Log-Euclidean Kernel . . . 107
4.5.3 The Wrapped Gaussian Kernel . . . 107
4.5.4 Numerical experiments comparing the kernels . . . 108
4.6 Conclusion . . . 109

4.1 Introduction

As we discussed in the previous chapter, the Gaussian distribution is often used to model random variables whose real distribution is not known. This is why, in Machine Learning (ML), the Gaussian distribution is commonly used as a foundational assumption for many algorithms and models. For example, the *Linear Discriminant Analysis* (LDA) (Section 4.3 of Hastie et al. 2009) is a classifier that assumes that each class k is modeled by a multivariate Gaussian $\mathcal{N}(\mu_k, \Sigma)$, each with their own mean μ_k and with a shared dispersion matrix Σ (see Figure 4.1). Then, to classify a new point, the likelihood of this new point under each Gaussian is computed, and the point is attributed to the class that maximizes this likelihood. The statistical model underneath the LDA is a *Gaussian Mixture Model* (GMM), a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions.

The main challenge for a GMM is to estimate the parameters of this mixture model, without knowing which point belongs to which component of the mixture. Other examples of ML models relying on the Gaussians are *Gaussian Processes*, a stochastic process made out of Gaussians that are used to solve regression and probabilistic classification problems, *Linear Regression*, where an essential assumption in order to have a closed-form formula of the regression coefficients is that the underlying data is Gaussian. More recently, *Variational AutoEncoders* (VAE) (Diederik and Max 2019), *diffusion models* or even *Flow Matching* (Lipman, R. T. Q. Chen, et al. 2023), use the Gaussian in different ways.

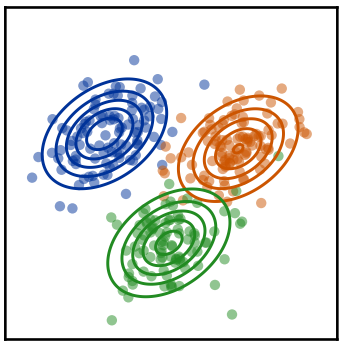


Figure 4.1. Example of an LDA. Each color represents a class, and a Gaussian $\mathcal{N}(\mu_k, \Sigma)$ is fitted for each class. The dispersion matrix Σ is common for all the classes.

Adapting classical algorithms to $\mathcal{P}_d$

Passing onto a curved Riemannian manifold, such as the manifold $\mathcal{P}_d$ of SPD matrices, the classical Euclidean Gaussian distribution cannot be used anymore as a prior to build ML algorithms. Hopefully, we have developed in the previous chapter a geometry-aware version of the multivariate Gaussian: the *wrapped Gaussian*. Building on top of that, one could transform all Euclidean learning algorithms to Riemannian algorithms by using the wrapped Gaussian to transform them into geometry-aware algorithms suited for $\mathcal{P}_d$. However, that would neglect all the work that has already been done to do learning on $\mathcal{P}_d$. Indeed, as we saw in Section 2.3, since the 2010s, researchers have started to develop learning methods adapted to curved Riemannian manifolds. They built these algorithms from scratch, adapting what works in a Euclidean setting to a manifold. Now that we have this wrapped Gaussian, we are able to see if these different learning algorithms can be reformulated using a Riemannian Gaussian prior. For this, we will also rely on two versions of the Riemannian Gaussian: the isotropic Gaussian on $\mathcal{P}_d$ introduced in Said, Bombrun, et al. (2017) and the wrapped Gaussian introduced in Chapter 3. We will see that we are able to build a framework for ML tools used on the manifold $\mathcal{P}_d$. More precisely, we will show how these two different probability distributions on $\mathcal{P}_d$ can be used to reinterpret classifiers, tools detecting outliers or performing dimension reduction on $\mathcal{P}_d$. We will show that all these tools can be united under the same probabilistic framework. This will give us an unexpected first answer to question 2: the wrapped Gaussian built in Chapter 3 is not only capable of building new learning tools for $\mathcal{P}_d$, it is also a convenient statistical model for rewriting already known learning tools. The wrapped Gaussian can thus be used to strengthen the theoretical underpinnings of existing tools.

Intrinsic versus Extrinsic

Moreover, in Section 2.3.6, we have grouped the state-of-the-art methods into two categories: the "intrinsic" and "extrinsic" methods. On the one hand, the intrinsic methods are defined directly on the manifold, without relying on any tangent space approximation of the manifold. On the other hand, extrinsic methods use classical Euclidean methods in a well-chosen tangent space, relying on the Euclidean structure of the latter. The different state-of-the-art methods reviewed in Section 2.3 were classified between these two categories in Table 2.4. An important example of an intrinsic method is the isotropic Gaussian while an example of an extrinsic method is the wrapped Gaussian. We will see that, unsurprisingly, the intrinsic learning algorithms will be closely related to the isotropic Gaussian, while the extrinsic ones will rely on the wrapped Gaussian. Furthermore, we will see that we are able to build new algorithms using the wrapped Gaussian, proposing anisotropic versions of isotropic algorithms but also

algorithms where the choice of the tangent space becomes an inherent parameter of the model and that can depend on the class when doing classification for example. This will give us a second answer to question 2.

Outline of the chapter

We will start by recalling, in Section 4.2, the two probability distributions of interest: the isotropic Gaussian and the wrapped Gaussian. In Section 4.3, popular classifiers on $\mathcal{P}_d$ will be reinterpreted using those distributions, as well as an outlier detection tool in Section 4.4 and Gaussian kernels in Section 4.5. We will also propose new algorithms that are based on the wrapped Gaussian proposed in Chapter 3, and we will lead several numerical experiments to compare the performances of the proposed algorithms to state-of-the-art Riemannian methods.

This probabilistic framework for the classifiers as well as the two newly proposed classifiers was published in:

> **Thibault de Surrel**, Fabien Lotte, Sylvain Chevallier, and Florian Yger (2025). "Wrapped Gaussian on the manifold of Symmetric Positive Definite Matrices". In: *Forty-second International Conference on Machine Learning*

For the full probabilistic framework, it was later published in:

> **Thibault de Surrel**, Florian Yger, Fabien Lotte, and Sylvain Chevallier (2025). "A probabilistic view on Riemannian machine learning models for SPD matrices". In: *International Conference on Geometric Science of Information*. Springer, pp. 142–151

We propose here an enhanced version of this probabilistic framework. Moreover, the study of Riemannian kernels is new, as well as the proposal of a new *Wrapped Gaussian Kernel*.

4.2 The two Gaussians of interest

Although discussed in depth in the previous chapter, for the sake of completeness of this chapter, we will start by recalling the definition of the two probability distributions of interest in this chapter: the isotropic Gaussian from Said, Bombrun, et al. (2017) and the wrapped Gaussian introduced in Chapter 3.

4.2.1 The isotropic Gaussian

The first Gaussian that we will use in the following is the isotropic Gaussian defined on the manifold $\mathcal{P}_d$ of SPD matrices from Said, Bombrun, et al. (2017). In the following, we will denote this distribution $G(\bar{\boldsymbol{X}}, \sigma^2)$ that depends on two parameters: $\bar{\boldsymbol{X}} \in \mathcal{P}_d$ which acts as the center of mass, and $\sigma > 0$ acting as the spread of the distribution. The density of this distribution with respect to the Riemannian volume is:

$$f^G_{\bar{\boldsymbol{X}},\sigma}(\boldsymbol{X}) = \frac{1}{\zeta(\sigma)} \exp\left(-\frac{\delta(\boldsymbol{X}, \bar{\boldsymbol{X}})^2}{2\sigma^2}\right), \tag{4.1}$$

where $\zeta(\sigma)$ is the normalizing constant. The authors also give an exact expression of $\zeta(\sigma)$ that does not depend on $\bar{\boldsymbol{X}}$, but only on σ. As $\sigma \in \mathbb{R}$, the spread is uniform in every direction of space, and the contours of this distribution are the level sets of the Riemannian distance function. This distribution is thus **isotropic**, and we will refer to it as the *isotropic Gaussian*. We also recall that using Proposition 7 of Said, Bombrun, et al. (2017), one can estimate the two parameters $\bar{\boldsymbol{X}}$ and σ of the distribution from an independent sample

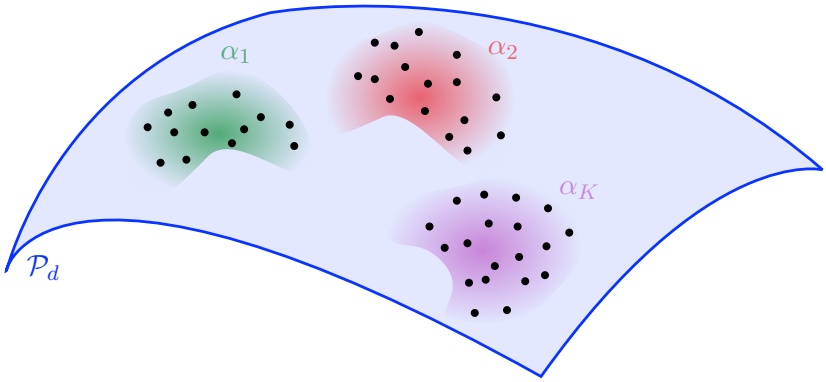


Figure 4.2. Illustration of the Maximum Likelihood Classifier. Each class is modeled using a distribution α_k. Then, to classify a new point $\boldsymbol{Z}$, the likelihood under each α_k is computed, and $\boldsymbol{Z}$ is classified based on the class maximizing the likelihood.

$\boldsymbol{X}_1, ..., \boldsymbol{X}_N \sim G(\bar{\boldsymbol{X}}, \sigma)$ using a Maximum Likelihood Estimator (MLE). More precisely, for $\bar{\boldsymbol{X}}$, the MLE $\hat{\bar{\boldsymbol{X}}}_N$ is the Riemannian mean of the sample (Moakher 2005), and for σ, the MLE is the only solution to the following non-linear equation in σ:

$$\sigma^3 \frac{\mathrm{d}}{\mathrm{d}\sigma} \log \zeta(\sigma) = \frac{1}{N} \sum_{i=1}^{N} \delta(\boldsymbol{X}_i, \hat{\bar{\boldsymbol{X}}}_N)^2.$$

4.2.2 The wrapped Gaussian

The second Gaussian distribution of interest in this chapter is the *wrapped Gaussian*. This distribution is the main focus of Chapter 3, thus, we will only recall the properties that we will need in what follows. The two main differences with the above isotropic Gaussian are that the wrapped Gaussian is **anisotropic** and that it relies on a tangent space in order to be defined. This distribution depends on three parameters: $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$ (we recall that $D = d(d+1)/2 = \dim \mathcal{P}_d$). Then, the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ is the push-forward of the Euclidean $\mathcal{N}(\mu, \Sigma)$ by the Riemannian exponential map composed with the vectorization of the tangent space $\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}$:

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = (\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}) \# \mathcal{N}(\mu, \Sigma).$$

The wrapped Gaussian corresponds to a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ in normal coordinates at $\boldsymbol{P}$. We recall that one is able to evaluate the density pointwise using Theorem 3.3. Moreover, we showed in Section 3.3, that the parameters $(\boldsymbol{P}, \mu, \Sigma)$ of the wrapped Gaussian can be estimated from a sample $\boldsymbol{X}_1, ..., \boldsymbol{X}_N \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ using the method of moments in the case of $\mu = 0$ *a priori* and using an MLE in a general setting where $\mu \neq 0$.

4.3 Classification under the Gaussian lens

In this section, we show that popular classification algorithms on $\mathcal{P}_d$ can be reinterpreted in the light of a probabilistic framework using the two different Gaussian distributions recalled in Section 4.2. Let us focus on the setup. We assume that we want to solve a multiclass classification problem on the manifold $\mathcal{P}_d$ of SPD matrices. Let us consider K balanced classes of labeled SPD matrices, and we denote by α_k the distribution modeling the k-th class. For simplicity, we will assume that each distribution α_k admits a density f_k with respect to the Riemannian volume. We illustrate this setup in Figure 4.2. In this setup, a natural classifier is the *Maximum Likelihood Classifier*:

Definition 4.1 (Maximum Likelihood Classifier)**.** Given K classes each modeled by a distribution α_k, a *Maximum Likelihood Classifier* assigns a new sample $\boldsymbol{Z}$ to the class k that maximizes the likelihood of $\boldsymbol{Z}$ under the distribution α_k:

$$\hat{y} = \arg\max_k f_k(\boldsymbol{Z}),$$

where f_k denotes the density of α_k.

In practice, one does not have access to the true distributions α_k, but only to samples $\boldsymbol{X}_1^k, ..., \boldsymbol{X}_{N_k}^k$ of each class. Thus, one starts by fitting the distributions α_k to the training data. This is done by estimating the parameters θ^k of a parametric class conditional density $f_k(\cdot \mid \theta^k)$, typically by fitting an MLE to each class separately on the training data. Then, at inference, one can evaluate $f_k(\boldsymbol{Z} \mid \hat{\theta}^k)$ for each class and pick the argmax. Let us start by giving an important example of a Maximum Likelihood Classifier that will be useful in the following: the Euclidean *Linear Discriminant Analysis* (LDA) and the *Quadratic Discriminant Analysis* (QDA):

Example 4.2 (The LDA and QDA)**.** The LDA classifier, in the Euclidean setting, is an instance of a Maximum Likelihood Classifier, where each class k is modeled by a multivariate Gaussian $\mathcal{N}(\mu^k, \Sigma)$ with a shared dispersion matrix Σ among the classes. First, as for any Maximum Likelihood Classifier, the parameters of each class are learned using an MLE on the training data. Then, to classify a new point z, the LDA compares the likelihood of z according to each class and chooses the class that has the highest log-likelihood. As the covariance matrix is shared among the classes, the decision boundaries are linear, which led to its name: *Linear* Discriminant Analysis (LDA). When one assumes that the covariance matrices are not equal among the classes, i.e. each class is modeled by $\mathcal{N}(\mu^k, \Sigma^k)$, the decision boundaries are quadratic. This classifier is called *Quadratic Discriminant Analysis* (QDA). We refer to Section 4.3 of Hastie et al. (2009) for more insights on the LDA and the QDA. One can restrict the covariance matrices to be diagonal, which leads to the *Diagonal LDA* and *Diagonal QDA* classifiers (Dudoit et al. 2002).

Our goal is now to show that popular classification algorithms on $\mathcal{P}_d$ can be interpreted as a Maximum Likelihood Classifier using the different Gaussian distributions recalled in Section 4.2. We will focus on the *Minimum Distance to Mean* (MDM) algorithm (Barachant, Bonnet, et al. 2010b), on the *Tangent Space Linear Discriminant Analysis* (TS-LDA) and on the *Tangent Space Quadratic Discriminant Analysis* (TS-QDA) algorithms (Barachant, Bonnet, et al. 2011). We will also introduce two new classifiers based on the wrapped Gaussian: the *Homogeneous Wrapped Discriminant Analysis* (Ho-WDA) and the *Heterogeneous Wrapped Discriminant Analysis* (He-WDA).

4.3.1 Minimum Distance to Mean (MDM)

The *Minimum Distance to Mean* (MDM) algorithm is a simple and popular classification algorithm for SPD matrices described in Barachant, Bonnet, et al. (2010b). Given a training set of SPD matrices, this algorithm estimates the Fréchet mean $\hat{\boldsymbol{X}}^k$ for each class $k \in \{1, ..., K\}$ and assigns a new sample $\boldsymbol{Z}$ to the class $\hat{k}$ that minimizes the Riemannian distance to the estimated mean $\hat{\boldsymbol{X}}^k$. We can reinterpret the MDM algorithm in the light of the isotropic Gaussian distribution defined in Section 4.2.1. Indeed, we have the following:

Proposition 4.3. *Let us suppose that each class is modeled by an isotropic Gaussian distribution centered at $\bar{\boldsymbol{X}}^k$ with a shared σ, i.e. $\alpha_k = G(\bar{\boldsymbol{X}}^k, \sigma^2)$, then, the MDM classifier is equivalent to a Maximum Likelihood Classifier.*

Proof. For a new unlabeled sample $\boldsymbol{Z} \in \mathcal{P}_d$, the decision $\hat{k}$ taken by the MDM algorithm is

$$\begin{aligned}\hat{k} = \arg\min_k \delta(\boldsymbol{Z}, \hat{\boldsymbol{X}}^k) &\iff \hat{k} = \arg\max_k -\frac{\delta(\boldsymbol{Z}, \hat{\boldsymbol{X}}^k)^2}{2\sigma^2} \\ &\iff \hat{k} = \arg\max_k \frac{1}{\zeta(\sigma)} \exp\left(-\frac{\delta(\boldsymbol{Z}, \hat{\boldsymbol{X}}^k)^2}{2\sigma^2}\right) \\ &\iff \hat{k} = \arg\max_k f^G_{\hat{\boldsymbol{X}}^k,\sigma}(\boldsymbol{Z}),\end{aligned}$$

where $f^G_{\hat{\boldsymbol{X}}^k,\sigma}$ is the density of the isotropic Gaussian described in Section 4.2.1. Moreover, as stated in Proposition 7 of Said, Bombrun, et al. (2017), the Fréchet mean $\hat{\boldsymbol{X}}^k$ is the MLE of the center of mass $\bar{\boldsymbol{X}}^k$ of the isotropic Gaussian $G(\bar{\boldsymbol{X}}^k, \sigma^2)$ of class k. Thus, the decision rule of the MDM is the same as the decision rule of the plug-in Maximum Likelihood Classifier where $\alpha_k = G(\hat{\boldsymbol{X}}^k, \sigma^2)$. □

Remark 4.4. Here, the spread σ does not play any role in the classification, so one only needs to estimate the Riemannian mean $\bar{\boldsymbol{X}}^k$ of each class. The classification is thus isotropic, it does not take into account the possibly non-uniform spread of the data.

The MDM is thus the first algorithm that fits into our new probabilistic framework for learning tools on $\mathcal{P}_d$. Moreover, in Section 2.3.6, the MDM was classified as an *intrinsic method*. This is coherent as it is based on the *intrinsic* isotropic Gaussian.

4.3.2 Tangent Space LDA/QDA

The *Tangent Space LDA* (TS-LDA) classifier is a generalization of the *Linear Discriminant Analysis* (LDA) classifier to the manifold of SPD matrices introduced in Barachant, Bonnet, et al. (2011). The idea is to first estimate the Fréchet mean $\hat{\boldsymbol{X}}_N$ of the whole training set, and then to project the training SPD matrices onto the tangent space $T_{\hat{\boldsymbol{X}}_N}\mathcal{P}_d$. Subsequently, these tangent vectors are vectorized into vectors of $\mathbb{R}^D$ using the vectorization $\text{Vect}_{\hat{\boldsymbol{X}}_N}$ given in Definition 2.31. One finally applies the classical Euclidean LDA algorithm, as described in Example 4.2, in $\mathbb{R}^D$. Similarly, the *Quadratic Discriminant Analysis* (QDA) can be extended to the *Tangent Space QDA* (TS-QDA).

These extrinsic algorithms deeply rely on a tangent space, and on a Gaussian distribution in this tangent space. Thus, we easily have a sense that the TS-LDA and TS-QDA are related to the wrapped Gaussian distribution described in Chapter 3. Indeed, one has the following proposition that we only write for the TS-LDA:

Proposition 4.5. *Let $\hat{\boldsymbol{X}}_N$ be the Fréchet mean of **all** the training points. Let $\hat{\mu}^k_N$ and $\hat{\Sigma}_N$ be the MLE of the Euclidean Gaussians in the tangent space at $\hat{\boldsymbol{X}}_N$, with a dispersion matrix $\hat{\Sigma}_N$ shared across the classes. Then, the decision rule of the TS-LDA is the same as the decision rule of the Maximum Likelihood Classifier where $\alpha_k = \text{WG}(\hat{\boldsymbol{X}}_N; \hat{\mu}^k_N, \hat{\Sigma}_N)$.*

Remark 4.6. A similar proposition can be written for the TS-QDA, by estimating a dispersion matrix $\hat{\Sigma}^k_N$ per class, and the distributions of each class in the Maximum Likelihood Classifier are now $\alpha_k = \text{WG}(\hat{\boldsymbol{X}}_N; \hat{\mu}^k_N, \hat{\Sigma}^k_N)$.

Proof. Given a new unlabeled sample $\boldsymbol{Z} \in \mathcal{P}_d$, the decision rule of the TS-LDA is

$$\begin{aligned}\hat{k} = \arg\max_k g_{\hat{\mu}^k_N,\hat{\Sigma}_N}(\text{Vect}_{\hat{\boldsymbol{X}}_N}(\text{Log}_{\hat{\boldsymbol{X}}_N}(\boldsymbol{Z}))) &\iff \hat{k} = \arg\max_k \frac{g_{\hat{\mu}^k_N,\hat{\Sigma}_N}(\text{Vect}_{\hat{\boldsymbol{X}}_N}(\text{Log}_{\hat{\boldsymbol{X}}_N}(\boldsymbol{Z})))}{|J_{\hat{\boldsymbol{X}}_N}(\text{Log}_{\hat{\boldsymbol{X}}_N}(\boldsymbol{Z}))|} \\ &\iff \hat{k} = \arg\max_k f^{WG}_{\hat{\boldsymbol{X}}_N;\hat{\mu}^k_N,\hat{\Sigma}_N}(\boldsymbol{Z}),\end{aligned}$$

where we recall that $g_{\hat{\mu}_N^k,\hat{\Sigma}_N}$ is the density of the Euclidean Gaussian $\mathcal{N}(\hat{\mu}_N^k,\hat{\Sigma}_N)$ and $f^{WG}_{\hat{\boldsymbol{X}}_N;\hat{\mu}_N^k,\hat{\Sigma}_N}$ is the density of the wrapped Gaussian $\mathrm{WG}(\hat{\boldsymbol{X}}_N;\hat{\mu}_N^k,\hat{\Sigma}_N)$. The Jacobian determinant $J_{\hat{\boldsymbol{X}}_N}(\mathrm{Log}_{\hat{\boldsymbol{X}}_N}(\boldsymbol{Z}))$ does not depend on k, thus, introducing it in the first equivalence does not affect the result. □

The TS-LDA and TS-QDA do then also fit in the probabilistic framework we are building. As expected since they are *extrinsic methods* based on the classification introduced in Section 2.3.6, they are based on the wrapped Gaussian. Let us make a few remarks on the above result.

Remark 4.7. In Proposition 4.5, the class distributions of the Maximum Likelihood Classifier at play are wrapped Gaussians. They are a specific type of wrapped Gaussian as they share the same anchor point $\hat{\boldsymbol{X}}_N$. The estimation of the parameters of these class-specific wrapped Gaussians does not follow the methodology introduced in Section 3.3: the anchor point is not estimated by maximum likelihood. Rather, in the TS-LDA (and in the TS-QDA), one first estimates the anchor point by the Fréchet mean $\hat{\boldsymbol{X}}_N$, as in Section 3.3.2, and only then, for this fixed anchor, the class-specific parameters $(\hat{\mu}_N^k,\hat{\Sigma}_N)$ using Proposition 3.35. This plug-in procedure differs from jointly maximizing

$$\prod_{k=1}^{K}\prod_{i=1}^{N_k} f_{\bar{\boldsymbol{X}};\mu^k,\Sigma}(\boldsymbol{X}_i^k)$$

over $(\bar{\boldsymbol{X}},\mu^1,\dots,\mu^K,\Sigma)$.

Remark 4.8. In Proposition 4.5, the anchor point $\hat{\boldsymbol{X}}_N$ is not only shared among the classes, it also constrains the estimated means $\hat{\mu}_N^k$. Indeed, one can do similar computations as in Section 3.3.2, by computing the Riemannian gradient of the Fréchet mean and using a first order optimality condition gives

$$\sum_{k=1}^{K}\sum_{i=1}^{N_k}\mathrm{Log}_{\hat{\boldsymbol{X}}_N}(\boldsymbol{X}_i^k)=0,$$

where the sum is taken over **all** the training points. Thus, by linearity of the vectorization $\mathrm{Vect}_{\hat{\boldsymbol{X}}_N}$, one has

$$\sum_{k=1}^{K}\sum_{i=1}^{N_k}\mathrm{Vect}_{\hat{\boldsymbol{X}}_N}(\mathrm{Log}_{\hat{\boldsymbol{X}}_N}(\boldsymbol{X}_i^k))=0_D,$$

and recognizing the expression of the MLE $\hat{\mu}_N^k$ given in Proposition 3.35, one gets $\sum_{k=1}^{K} N_k\hat{\mu}_N^k = 0_D$, that is

$$\sum_{k=1}^{K}\hat{\mu}_N^k = 0_D,$$

as the classes are balanced. The TS-LDA can therefore be seen as a *constrained* wrapped Gaussian model: all the classes share the same anchor point, and their means are forced to sum to zero. This is one of the motivations for the classifiers that we will introduce in the next section, where each class is given its own anchor point.

4.3.3 Wrapped Discriminant Analysis

We have shown how some of the most popular Riemannian classifiers used on the manifold $\mathcal{P}_d$ of SPD matrices, the MDM, the Tangent-Space LDA and the Tangent-Space QDA fit together into a global probabilistic framework. Now, we want to introduce two new classifiers based

Classifier	α_k	Anchor point	Mean μ_k	Covariance
MDM	$G(\bar{\boldsymbol{X}}^k, \sigma^2)$	C	—	S
TS-LDA	$\mathrm{WG}(\bar{\boldsymbol{X}}^*; \mu^k, \Sigma)$	S	C	S
TS-QDA	$\mathrm{WG}(\bar{\boldsymbol{X}}^*; \mu^k, \Sigma^k)$	S	C	C
Ho-WDA	$\mathrm{WG}(\boldsymbol{P}^k; \mu^k, \Sigma)$	C	C	S
He-WDA	$\mathrm{WG}(\boldsymbol{P}^k; \mu^k, \Sigma^k)$	C	C	C

Table 4.1. Classifiers reinterpreted as Maximum Likelihood Classifiers. "S" denotes a parameter shared across all classes while "C" denotes a class-specific parameter. The bottom two rows are the new classifiers proposed in Section 4.3.3.

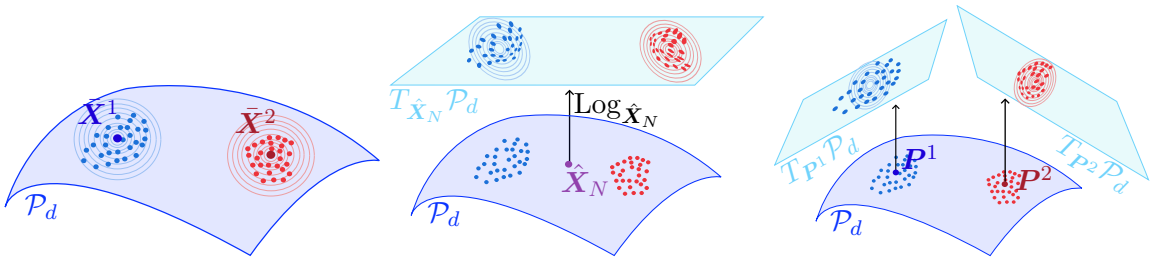
on the wrapped Gaussian. These classifiers are heavily inspired by the LDA/QDA and their tangent space counterparts. Indeed, as described in Proposition 4.5, the Tangent-Space LDA is based on a wrapped Gaussian with a shared anchor among the classes, that is estimated using the Fréchet mean. We can now imagine that this anchor point is not shared among the classes, but rather specific to each class. This gives rise to new classifiers that project each class onto its own tangent space. We call such classifiers *Wrapped Discriminant Analysis* (WDA). As for the LDA and QDA, we propose two versions of WDA:

1. First, let us model each class by a wrapped Gaussian with a shared covariance matrix Σ among the classes:
$$\alpha_k = \mathrm{WG}(\boldsymbol{P}^k; \mu^k, \Sigma).$$
We call this classifier the *Homogeneous Wrapped Discriminant Analysis* (Ho-WDA). To learn the parameters of each class, we optimize an MLE on the whole model to find the parameters:
$$(\boldsymbol{P}^1, ..., \boldsymbol{P}^K, \mu^1, ..., \mu^K, \Sigma) \in \underset{\boldsymbol{P}^k, \mu^k, \Sigma}{\arg\max} \prod_{k=1}^{K} \prod_{i=1}^{N_k} f_{\boldsymbol{P}^k; \mu^k, \Sigma}(\boldsymbol{X}_i^k)$$
where $(\boldsymbol{X}^k)_{i=1,..,N_k}$ are the training points of class k. The implementation of this MLE is similar as the one done in Section 3.3.
2. As for the QDA, we propose another version of this classifier where each class has its own covariance matrix Σ^k:
$$\alpha_k = \mathrm{WG}(\boldsymbol{P}^k; \mu^k, \Sigma^k).$$
We call this classifier the *Heterogeneous Wrapped Discriminant Analysis* (He-WDA). In that case, an MLE is optimized on each class individually, as in Section 3.3:
$$\forall k \in \{1, ..., K\}, \ (\boldsymbol{P}^k, \mu^k, \Sigma^k) \in \underset{\boldsymbol{P}, \mu, \Sigma}{\arg\max} \prod_{i=1}^{N_k} f_{\boldsymbol{P}; \mu, \Sigma}(\boldsymbol{X}_i^k).$$

Then, at inference time, for both classifiers, we compute the likelihood of a new unlabeled sample under each α_k, and classify it based on the class that maximizes the likelihood.

The hope for this new classifier is that it will have more flexibility as it has more degrees of freedom. It will therefore be able to better estimate the true distribution of the data and this will lead to better classification results. Table 4.1 summarizes all classifiers introduced in this section and Figure D.3 illustrates them. Those 5 classifiers will be compared in a numerical experiment in Section 4.3.5.

> *Remark* 4.9. Apart from the fact that each class now has its own specific anchor point $\boldsymbol{P}^k$, the Wrapped Discriminant Analysis classifiers proposed here also solve the estimation issue of the TS-LDA/QDA mentioned in Remark 4.7 and Remark 4.8. Indeed, here, we use the



(a) MDM. **(b)** TS-LDA. **(c)** Ho-WDA.

Figure 4.3. Illustration of the different Riemannian classifiers considered here on a two class problem: MDM models each class k by an isotropic Gaussian centered at its own Fréchet mean $\bar{\boldsymbol{X}}^k$. TS-LDA maps every class to a single common tangent space $T_{\hat{\boldsymbol{X}}_N}\mathcal{P}_d$, at the Fréchet mean $\hat{\boldsymbol{X}}_N$ estimated on all training points. Ho-WDA instead gives each class its own anchor $\boldsymbol{P}^k$ and its own tangent space $T_{\boldsymbol{P}^k}\mathcal{P}_d$.

full MLE to estimate the parameters of the wrapped Gaussians in play. We do not rely on the Fréchet mean to estimate the anchor of the different classes as the TS-LDA/QDA do. If we were to use the Fréchet mean, we would fall back into the problems mentioned in Section 3.3.2: the estimated means $\hat{\mu}_N^k$ in the tangent spaces would always be 0_D, thus the model would once again be restricted to a narrower class of wrapped Gaussians.

4.3.4 Other probabilistic classifiers for SPD matrices

In this chapter, we mainly focus on the different Gaussian distributions described in Section 4.2. However, other authors have studied Maximum Likelihood Classifiers based on other probability distributions. For instance, in Ayadi et al. (2024), the authors propose a Maximum Likelihood Classifier based on the Elliptical Wishart distribution. They also adapt it to an unsupervised clustering scenario leveraging the K-means clustering algorithm. In Ayadi et al. (2023), the authors show that if the metric used in the MDM algorithm is based on the Kullback-Leibler divergence between two centered multivariate Gaussian distributions, then, the MDM algorithm is a Maximum Likelihood Classifier based on the t-Wishart distribution.

4.3.5 Numerical experiments comparing the classifiers

In this section, we want to compare the two classifiers we proposed: the Ho-WDA and the He-WDA to the other classifiers: the MDM, the Tangent Space LDA (TS-LDA) and the Tangent Space QDA (TS-QDA). For this, we ran some numerical experiments on 8 different real datasets coming from several applications. Let us start by describing them, then give details on the methodology used and finally, give the results.

The datasets For this experiment, we used 8 different datasets that are summarized in Table 4.2. Let us give more details on them:

- **BCI Datasets**: We considered 2 datasets from Brain-Computer Interfaces (BCI) for our experiments: *BNCI2014004* (Leeb et al. 2007) and *Zhou2016* (B. Zhou et al. 2016). They consist of several subjects and several sessions per subject doing a Motor Imagery task (G. Pfurtscheller and C. Neuper 2001). We used the library MOABB (Aristimunha, Carrara, et al. 2026) to load and preprocess the data. For each EEG, we start by applying a standard band-pass filter with range $[7; 35]$ Hz. Then, we used the Ledoit-Wolf shrunk covariance matrix (Ledoit and Wolf 2004) in order to estimate the covariance matrices and to avoid ill-conditioned matrices. Here, we consider only two classes: `right_hand` and `left_hand`.

Dataset	Application	Dimension	# matrices	# classes
BNCI2014004	BCI	3×3	720×9	2
Zhou2016	BCI	5×5	320×4	2
Air Quality	Atmospheric data	6×6	102	3
Indiana Pines	Hyperspectral imaging	5×5	14,641	12
Pavia Univ.	Hyperspectral imaging	5×5	185,176	6
Salinas	Hyperspectral imaging	5×5	94,184	17
Textile	Image analysis	10×10	16,000	2
BreizhCrops	Multispectral imaging	13×13	177,658	6

Table 4.2. Summary of the datasets used for the experiments comparing the classifiers.

- **AirQuality**: This dataset is from the Beijing Municipal Monitoring Center. It is a dataset of air quality monitored from 34 different sites in Beijing, China (Hua et al. 2021). For each site, six atmospheric pollutants were recorded every hour: CO, NO_2, O_3, PM_{10}, $PM_{2.5}$ and SO_2. We used the same preprocessing as in Smith, Hua, et al. (2022) to get a point cloud of 102 covariance matrices of size 6×6, representing each the covariances between the six different pollutants given above. Each covariance matrix has a label depending on which period it represents: weekdays, weekends or holidays.
- **Indiana, Pavia Uni, Salinas**[1]. Each dataset contains one hyperspectral image of a certain region containing a unique number of reflectance bands. We applied the same preprocessing as in Collas, Bouchard, et al. (2021) or Bouchard et al. (2024) that consists in four main steps. First, we normalize the data by subtracting the image global mean. Then, we apply a PCA to reduce the dimension of the data to 5. A sliding window with no overlap is then used around each pixel for data sampling and then vectorized. In our experiments, we used a window of size 25×25. Finally, we compute the covariance matrix of each vectorized window using the Sample Covariance Matrix (see Equation (2.1)) to get a point cloud of covariance matrices. For each covariance matrix, its class was computed by taking the majority class of the pixels in the window.
- **Textile**: This dataset is made of a set of real images from textile manufacturing that contain non-defective and defective woven textiles. These images come from the public MVTec Anomaly Detection dataset (Bergmann et al. 2021). The same preprocessing as in Smith, Laubach, et al. (2022) was applied to get the covariance matrices: each image is first normalized, then a bank of $d = 10$ filter responses is computed at every pixel: the (normalized) intensity itself, Gaussian smoothings at different scales, and Laplacian and Farid (gradient) filters applied both to the raw image and to two of the smoothed versions. After stacking these responses over the pixels, the Sample Covariance Matrix (see Equation (2.1)) of the pixel-wise features is computed for each image. The two classes correspond to defective and non-defective textiles.
- **BreizhCrops**: This dataset is intended for supervised classification of field crops from satellite time series. We used the dataset `FRH01` that is composed of satellite time series from the French region *Finistère*. As they are multivariate time-series, we simply converted them to covariance matrices using the Oracle Approximating Shrinkage estimator (Y. Chen et al. 2010). The classes correspond to different types of crops. More details on this dataset can be found in the original paper (Rußwurm et al. 2020).

[1]These three hyperspectral remote sensing datasets are all publicly available at https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes.

Dataset	MDM	TS-LDA	TS-QDA	Ho-WDA	He-WDA
BNCI2014004	78.71$_{\pm 14.53}$	**78.73**$_{\pm 14.52}$	76.07$_{\pm 13.94}$	75.38$_{\pm 14.40}$	74.37$_{\pm 14.73}$
Zhou2016	91.18$_{\pm 5.51}$	**91.21**$_{\pm 5.50}$	89.45$_{\pm 7.43}$	85.92$_{\pm 9.20}$	82.86$_{\pm 11.65}$
Air Quality	94.05$_{\pm 6.53}$	94.05$_{\pm 6.53}$	**97.05**$_{\pm 4.45}$	96.05$_{\pm 4.17}$	**97.00**$_{\pm 4.47}$
Indiana	58.01$_{\pm 0.72}$	67.07$_{\pm 0.53}$	73.38$_{\pm 0.45}$	73.74$_{\pm 0.51}$	**74.30**$_{\pm 0.83}$
Pavia Uni.	72.32$_{\pm 0.59}$	84.61$_{\pm 0.06}$	87.16$_{\pm 0.09}$	**87.36**$_{\pm 0.07}$	85.54$_{\pm 0.15}$
Salinas	36.42$_{\pm 0.12}$	46.30$_{\pm 0.17}$	69.87$_{\pm 0.21}$	**71.20**$_{\pm 0.33}$	62.39$_{\pm 0.22}$
Textile	83.08$_{\pm 0.62}$	83.12$_{\pm 0.63}$	86.03$_{\pm 0.66}$	**86.26**$_{\pm 0.59}$	85.93$_{\pm 0.77}$
BreizhCrops	45.48$_{\pm 0.25}$	47.67$_{\pm 0.32}$	50.72$_{\pm 0.28}$	**54.66**$_{\pm 0.45}$	51.33$_{\pm 0.58}$

Table 4.3. Classification accuracy (%, mean ± standard deviation) of the different classifiers on the benchmark datasets. The best result for each dataset is highlighted in bold.

The methodology For this experiment, we relied on the classifiers MDM, TS-LDA and TS-QDA implemented in pyRiemann (Barachant, Barthélemy, et al. 2026) and Scikit-learn (Pedregosa et al. 2011)[2]. For the Ho-WDA and He-WDA, we implemented them using our MLE to estimate the parameters of the wrapped Gaussians (as described in Section 3.3). To optimize the MLE, we used in practice the Riemannian Conjugate Gradient method (Boumal 2023) with a maximum of $1,000,000$ iterations and a max time set to 2 hours. For this experiment, we restricted ourselves to the case where the dispersion matrices Σ are diagonal. Indeed, we saw at Section 3.3 that, when the dispersion matrix Σ was full, the MLE for the wrapped Gaussian required a lot of points in order to converge. As some of our datasets only have a few hundred points, restricting Σ to be only diagonal would lead to better estimation results, and thus, a better fitted model.

For the BCI datasets (BNCI2014001 and Zhou2016), the experiment we ran was cross-subject: each classifier was trained on all subject except one and tested on this last subject. For the non-BCI datasets (AirQuality, Indiana, Pavia Uni, Salinas, Textile and BreizhCrops), we used a 5-fold cross-validation to evaluate the performance of the classifiers.

The results We give the accuracy of the classifiers we study on the different datasets in Table 4.3. We can see two different behaviors. First, on the datasets with a lot of matrices (more than $10,000$ so the datasets Textile, Salinas, Indiana Pines, Pavia Univ., BreizhCrops), the Ho-WDA and He-WDA perform the best. The number of parameters to estimate is high, so the more samples one has, the better the estimation will be as we illustrated in the synthetic experiments in Section 3.3. For BreizhCrops, even if the matrices are of size 13×13, we have a lot of points ($177,658$) so the MLE is able to correctly estimate all the parameters. Secondly, on the BCI datasets (BNCI2014004 and Zhou2016), we have fewer points (around $5,000$ training points for BNCI2014004 and less than $1,000$ for Zhou2016) so the estimation of the parameters of the underlying wrapped Gaussians is less precise. In this case, one can see that the simpler models MDM and TS-LDA perform the best where the wrapped Gaussian based models Ho-WDA and He-WDA perform less well. However, on the AirQuality dataset, the Ho-WDA and He-WDA perform the best. This is interesting as the number of matrices available in the dataset is small (102). An explanation could be that the underlying distribution of the data is not very complex, so a few points are enough to correctly estimate them. In BCI datasets, the distribution is more complex as we are performing a cross-subject task, so one needs more points to correctly estimate the distribution. Finally, we do not observe a clear dominance of the He-WDA over the Ho-WDA. This is similar to the

[2]The TS-LDA and TS-QDA were implemented by using the `TangentSpace` class from pyRiemann and the LDA/Naive Bayes classifier from Scikit-learn.

difference between the LDA and QDA where, often, an LDA can correctly classify data. In theory, the Ho-WDA should be a special case of the He-WDA where the covariance matrices for each class are the same. However, in practice, as the number of points per class can be small and as the number of parameters to estimate is higher for the He-WDA, the estimation of the covariance matrices per class can be noisy and the performance of the He-WDA can be worse than that of the Ho-WDA. For example, for the dataset Salinas, some classes have only a few hundred samples, which is not enough to estimate the covariance matrix accurately. The Ho-WDA, on the other hand, estimates a single covariance matrix for all the classes and is less sensitive to the number of samples per class. A similar comment could be made regarding TS-LDA and Ho-WDA (or TS-QDA and Ho-WDA): TS-LDA is a special case of Ho-WDA where the anchor points of each class are the same. However, we observe that on the BCI datasets, thus complex distribution, a simpler model will perform better.

4.4 Outlier detection *à la* Gaussian

When dealing with real data, such as EEGs in our case, before even thinking of classifying them, a first crucial preprocessing task is to detect outliers. Outliers are, as stated in Chandola et al. (2009): "patterns in data that do not conform to a well-defined notion of normal behavior". They must be detected early in the pipeline, as keeping them could cause some noise in the learning procedure. In Brain-Computer Interfaces (BCI), EEGs are subject to a lot of artifacts, and we have already described some techniques to detect outliers in Section 2.1.1, such as *Independent Component Analysis* (ICA, Hyvärinen and Oja 2000) or the *Signal-Space Projection* (SSP, Uusitalo and Ilmoniemi 1997). These techniques are often used offline, that is, when the whole dataset has been recorded. However, it would be useful to have an online tool, that is capable of rejecting EEGs as they are recorded. One of the first works to tackle the online outlier detection problem using the Riemannian framework was the *Riemannian potato* introduced in Barachant, Andreev, et al. (2013).

The *Riemannian potato* is a tool used to automatically detect outliers in the data. The idea of the Riemannian potato is to estimate a reference SPD matrix and a measure of dispersion (z-score), and then to reject all SPD matrices that are too far from the reference matrix. Let $(\boldsymbol{X}_i)_{i=1,\dots,N}$ be a set of SPD matrices, and let us denote $\bar{\boldsymbol{X}}$ the reference SPD matrix. We want to estimate the dispersion of a new matrix $\boldsymbol{Z} \in \mathcal{P}_d$ from the set $(\boldsymbol{X}_i)_{i=1,\dots,N}$ that is represented by $\bar{\boldsymbol{X}}$. For this, the z-score z of $\boldsymbol{Z}$ is computed as:

$$z = \frac{\delta(\boldsymbol{Z}, \bar{\boldsymbol{X}}) - \mu}{\sigma},$$

where one has $\mu = \frac{1}{N}\sum_{i=1}^{N} \delta(\boldsymbol{X}_i, \bar{\boldsymbol{X}})$ and $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\delta(\boldsymbol{X}_i, \bar{\boldsymbol{X}}) - \mu)^2}$. Provided a threshold z_{th}, the matrix $\boldsymbol{Z}$ is accepted if, and only if, $z \leqslant z_{\text{th}}$. In the original paper Barachant, Andreev, et al. (2013), this threshold z_{th} is chosen as $z_{\text{th}} = \mu + 2.5\sigma$. An example of a Riemannian potato is given in Figure 4.4. The Riemannian potato is a simple yet efficient tool to detect outliers in the data. Using the isotropic Gaussian distribution defined in Section 4.2.1, we can see that the Riemannian potato rejects points that are too unlikely under a specific isotropic Gaussian:

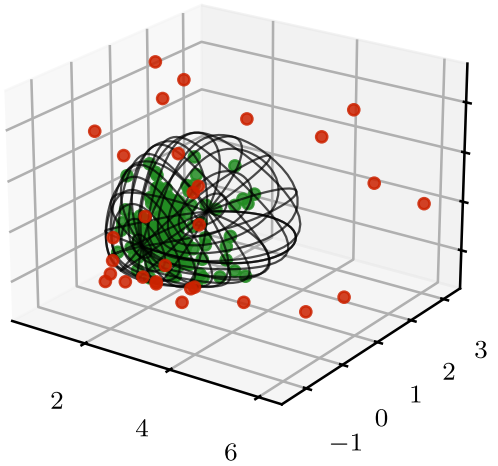


Figure 4.4. Illustration of a Riemannian potato on the set $\mathcal{P}_2$ of 2×2 SPD matrices. Green (resp. red) markers indicate matrices lying inside (resp. outside) the potato.

Proposition 4.10. *Let $\bar{\boldsymbol{X}} \in \mathcal{P}_d$ be a reference matrix, $\mu \in \mathbb{R}, \sigma > 0$ and $z_{th} > 0$. Then, $\boldsymbol{Z} \in \mathcal{P}_d$ is accepted by the Riemannian potato with threshold z_{th} if and only if*

$$\mathcal{L}_{\bar{\boldsymbol{X}},\sigma}(\boldsymbol{Z}) \geqslant \ell_{th},$$

where $\mathcal{L}_{\bar{\boldsymbol{X}},\sigma}$ denotes the likelihood of $G(\bar{\boldsymbol{X}}, \sigma)$ and where $\ell_{th} = \frac{1}{\zeta(\sigma)} \exp\left(-\frac{1}{2}\left(z_{th} + \frac{\mu}{\sigma}\right)^2\right)$.

Proof. The new matrix $\boldsymbol{Z}$ is accepted by the Riemannian potato if, and only if $z \leqslant z_{\text{th}}$. Moreover, one has that

$$\begin{aligned}
z \leqslant z_{\text{th}} &\iff \frac{\delta(\boldsymbol{Z}, \bar{\boldsymbol{X}}) - \mu}{\sigma} \leqslant z_{\text{th}} \\
&\iff \frac{\delta(\boldsymbol{Z}, \bar{\boldsymbol{X}})^2}{\sigma^2} \leqslant \left(z_{\text{th}} + \frac{\mu}{\sigma}\right)^2 \\
&\iff \exp\left(-\frac{1}{2}\left(z_{\text{th}} + \frac{\mu}{\sigma}\right)^2\right) \leqslant \exp\left(-\frac{\delta(\boldsymbol{Z}, \bar{\boldsymbol{X}})^2}{2\sigma^2}\right) \\
&\iff \frac{1}{\zeta(\sigma)} \exp\left(-\frac{1}{2}\left(z_{\text{th}} + \frac{\mu}{\sigma}\right)^2\right) \leqslant \mathcal{L}_{\bar{\boldsymbol{X}},\sigma}(\boldsymbol{Z}).
\end{aligned}$$

Then, denoting $\ell_{\text{th}} = \frac{1}{\zeta(\sigma)} \exp\left(-\frac{1}{2}\left(z_{\text{th}} + \frac{\mu}{\sigma}\right)^2\right)$, we have the result. Note that, to go from the first to the second line, all the quantities are positive, we can thus apply the square without any problems. □

This result is quite coherent: when computing a z-score, the goal is to try to know how far the new point $\boldsymbol{Z}$ is from the reference point $\bar{\boldsymbol{X}}$. Then, if it is sufficiently far, one can discard it. This is precisely the question addressed by the likelihood: how likely is it that the observation $\boldsymbol{Z}$ was generated by the isotropic Gaussian $G(\bar{\boldsymbol{X}}, \sigma^2)$? If the likelihood is too small, then one can treat it as an outlier.

Remark 4.11. An important remark on this method is that it is **isotropic**, it treats every direction in space equally. It compares the distance of a new point $\boldsymbol{Z}$ to the level sets of the distance function to the reference point $\bar{\boldsymbol{X}}$ and those level sets form (Riemannian) sphere around centered at $\bar{\boldsymbol{X}}$. One could want to have an anisotropic method, that learns how to form ellipsoid around $\bar{\boldsymbol{X}}$ instead of spheres. One could imagine switching the isotropic Gaussian in Proposition 4.10 with a wrapped Gaussian defined in Chapter 3. This would lead to an anisotropic *wrapped potato*. We will propose and study in more depth such a wrapped potato in Chapter 6.

Remark 4.12. In practice, the Riemannian potato was used as an *online* outlier detection tool, thus it was built to detect outliers as new EEGs would be recorded. In order to do that, the reference matrix was the Fréchet mean estimated iteratively as new samples $\boldsymbol{X}_i$ were added to the dataset. Then, each new sample was first tested against the current Riemannian potato, and if it was accepted, it was added to the dataset. The parameters $\bar{\boldsymbol{X}}$, μ and σ where then re-estimated on the new dataset and the process could continue.

The Riemannian potato has been improved in Barthélemy et al. (2019), where the authors changed the way of computing the z-score by relying on geometric statistics, improved the offline initialization and adapted it to do transfer learning. Their main contribution was to also introduce a *Riemannian potato field*, that computes different band-pass filter and estimate a Riemannian potato for each of those filters. They then are able to combine the z-score of those different potatoes in order to take a more informed decision.

Another Riemannian method to detect a change in the distribution is the *Riemannian Online Change Point Detection* algorithm proposed in X. Wang, R. A. Borsoi, et al. (2024)

and later improved in X. Wang, R. Borsoi, et al. (2025). This algorithm estimates two online Fréchet means with two different learning rates (or step sizes) and compare the distances between these two estimated means. If the distance is over a fixed threshold, it will flag the time step as a change point. The intuition is that a large learning rate leads the mean to adapt quickly to new data, while a smaller learning rate has a longer memory. Thus, if a brutal change occurs in the data, the mean with a large learning rate will rapidly capture it, while the other mean will take more time to adapt. These methods do not fit in our probabilistic framework.



4.5 Kernels with a Gaussian flavor

As we described in Section 2.3.2, kernel methods (Hofmann et al. 2008; Cuturi 2009) can be very useful when the data is intrinsically non-linear, as the kernel trick gives a way to leverage simple linear classifier for non-linear problems. These methods have been studied specifically for structured data (Gartner 2008) and for covariance matrices for BCI in Yger (2013) and Wagner vom Berg (2025). They rely deeply on *kernels*, i.e. a similarity function over all pairs of data points. The selection of the best kernel is a crucial choice as it will guide the similarity structure and implicit feature space one will impose on the data. A crucial property of kernel k is that it should be *positive semidefinite* i.e. for all $N \in \mathbb{N}^*, x_1, ...x_N \in \mathcal{X}$ and $c_1, ..., c_N \in \mathbb{R}$, where $\mathcal{X}$ is the space of the input data, the kernel k must verify:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} c_i c_j k(x_i, x_j) \geqslant 0.$$

Positive semidefiniteness is what guarantees that the kernel trick actually works, and then, the whole kernel-method machinery can be deployed (see Section 2.2 of Hofmann et al. 2008).

In a Euclidean setting, one of the most popular kernel used is the Gaussian kernel, that is defined as:

$$k(x, y) = \exp\left(-\frac{\|x - y\|_2^2}{2\sigma^2}\right) \quad \sigma > 0. \tag{4.2}$$

This kernel is a special instance of a radial basis function kernels, as it satisfies the property: $k(x, y) = \tilde{k}(\|x - y\|)$. Moreover, this kernel is precisely connected to the Euclidean Gaussian distribution as it proportional to the Gaussian density centered at y and evaluated at x. We have already reviewed some Riemannian kernels in Section 2.3.2 and the goal now is similar as above: connect the Riemannian kernels on $\mathcal{P}_d$ to the two Riemannian Gaussians described in Section 4.2 and see how the wrapped Gaussian can be used to build new kernels.

4.5.1 The AIRM Kernel

The first kernel that would come to mind when wanting to build a Gaussian kernel on the manifold $\mathcal{P}_d$ inspired by the Euclidean Gaussian kernel given in Equation (4.2) is to replace the Euclidean distance $\|x - y\|$ by the AIRM distance $\delta(\boldsymbol{X}, \boldsymbol{Y})$ as defined in Definition 2.13. This would lead to a Riemannian Gaussian kernel as the one given in Equation (2.8) (for $q = 2$ and $\lambda = \sigma^{-2}$):

$$k_{\text{AIRM}}(\boldsymbol{X}, \boldsymbol{Y}) = \exp\left(-\frac{\delta(\boldsymbol{X}, \boldsymbol{Y})^2}{2\sigma^2}\right), \quad \sigma > 0.$$

This kernel is thus exactly proportional to the density of the isotropic Gaussian $G(\boldsymbol{Y}, \sigma)$ centered at $\boldsymbol{Y}$ and evaluated at $\boldsymbol{X}$. It is the exact counterpart of the Euclidean Gaussian kernel.

However, as we have seen in Theorem 2.34, this kernel is not positive definite for all $\sigma > 0$. Indeed, we recall this theorem from Feragen et al. (2015):

Theorem 4.13. *Let $\mathcal{M}$ be a complete, smooth Riemannian manifold with its associated Riemannian distance δ. Then, if the Gaussian kernel $k(\boldsymbol{X}, \boldsymbol{Y}) = \exp\left(-\lambda\delta(\boldsymbol{X}, \boldsymbol{Y})^2\right)$ is positive definite for all $\lambda > 0$, then the Riemannian manifold $\mathcal{M}$ is isometric to a Euclidean space.*

However, we know from Section 2.2.3.1 that $\mathcal{P}_d$ equipped with the AIRM geometry is curved (see Proposition 2.21), thus it is not isometric to a Euclidean space. Thus, in order to use this Riemannian Gaussian kernel in practice, one needs to correctly tune the parameter σ^2 (or equivalently λ).

4.5.2 The Log-Euclidean Kernel

According to the above Theorem 4.13, if we want to build a Gaussian kernel on the manifold $\mathcal{P}_d$, we must eliminate all metrics leading to a curved space. A way of doing this is to consider the log-Euclidean geometry of $\mathcal{P}_d$, described in Section 2.2.3.2. Indeed, $\mathcal{P}_d$ equipped with this metric is a flat space. In Jayasumana et al. (2013), the authors proposed the following log-Euclidean kernel on $\mathcal{P}_d$:

$$k_{\text{LE}}(\boldsymbol{X}, \boldsymbol{Y}) = \exp\left(-\frac{\|\log(\boldsymbol{X}) - \log(\boldsymbol{Y})\|_F^2}{2\sigma^2}\right) \quad \sigma > 0. \tag{4.3}$$

This is exactly the Gaussian log-Euclidean kernel as, referring to Proposition 2.26, the log-Euclidean distance between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is $\|\log(\boldsymbol{X}) - \log(\boldsymbol{Y})\|_F$. Moreover, Jayasumana et al. (2013) proved (see their Corollary 4.5) that this kernel is positive definite for all $\sigma > 0$. One can rewrite this log-Euclidean distance as, using the fact that the vectorization Vect_{I_d} is an isometry between $(T_{I_d}\mathcal{P}_d, \langle\cdot,\cdot\rangle_F)$ and $(\mathbb{R}^D, \langle\cdot,\cdot\rangle_2)$ from Proposition 2.32:

$$\|\log(\boldsymbol{X}) - \log(\boldsymbol{Y})\|_F = \|\text{Vect}_{I_d}(\text{Log}_{I_d}(\boldsymbol{X})) - \text{Vect}_{I_d}(\text{Log}_{I_d}(\boldsymbol{Y}))\|_2.$$

Plugging this expression into the expression of the log-Euclidean kernel given in Equation (4.3) sparks two remarks:

Remark 4.14. This log-Euclidean kernel resembles a lot the density of a wrapped Gaussian with anchor point $\boldsymbol{P} = I_d$, a mean in the tangent space $\mu = \text{Vect}_{I_d}(\text{Log}_{I_d}(\boldsymbol{Y}))$ and a dispersion matrix $\Sigma = \sigma^{-2} I_D$ evaluated at $\boldsymbol{X}$. However, the main difference is that this log-Euclidean kernel does not take into account the Jacobian determinant J_{I_d} of the exponential mapping. Thus, it treats the embedding $\text{Vect}_{I_d} \circ \text{Log}_{I_d}$ as flat. This is coherent as the log-Euclidean geometry is flat.

Remark 4.15. This log-Euclidean kernel relies on the Riemannian logarithm $\text{Log}_{I_d} = \log$ at the identity matrix, thus mapping all the points on the tangent space $T_{I_d}\mathcal{P}_d$ at the identity. Choosing the identity as the reference point can lead to a potential deformation of the data if the data is not centered around the identity (Jaquier, Rozo, and Asfour 2024). As remarked in Remark 2.28, this default choice of the identity matrix can be mitigated by choosing another reference point (for example the Fréchet mean of the dataset) and using the log-Euclidean metric associated to this point. Another (equivalent) solution could be to center the data around the identity. This could help better localize the tangent space.

Those two remarks have led us to the definition of a new Riemannian kernel on $\mathcal{P}_d$, based on the wrapped Gaussian and that is described in the following section.

4.5.3 The Wrapped Gaussian Kernel

As we did for the classifiers in Section 4.3 by proposing the *Wrapped Discriminant Analysis* classifiers in Section 4.3.3, we now propose a new kernel based on the wrapped Gaussian that

could be used on the manifold $\mathcal{P}_d$. This kernel will answer the remarks made above on the log-Euclidean kernel. We call this new kernel the *Wrapped Gaussian Kernel*. This kernel depends on two parameters: an anchor point $\boldsymbol{P} \in \mathcal{P}_d$ and a scaling parameter $\sigma > 0$.

Definition 4.16 (Wrapped Gaussian Kernel)**.** Let $\boldsymbol{P} \in \mathcal{P}_d$ and $\sigma > 0$. We define the *Wrapped Gaussian Kernel* k_{WG} as follows:

$$k_{\mathrm{WG}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{\sqrt{J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X})) \cdot J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y}))}} \times \exp\left(-\frac{\| \mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X})) - \mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y})) \|_2^2}{2\sigma^2}\right) \tag{4.4}$$

where $J_{\boldsymbol{P}}$ denotes the Jacobian determinant of the exponential map $\mathrm{Exp}_{\boldsymbol{P}}$.

This Wrapped Gaussian Kernel is proportional to the density of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mathrm{Vect}_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y})), \sigma^2 I_D)$ evaluated at $\boldsymbol{X}$, symmetrized via the geometric mean of the two Jacobian factors to make the kernel symmetric. As for the previous log-Euclidean kernel, one has the following proposition:

Proposition 4.17. *The wrapped Gaussian kernel given in Definition 4.16 is positive definite for all $\sigma > 0$.*

Proof. The wrapped Gaussian kernel of Definition 4.16 can be rewritten as the Hadamard product of the symmetric rank-one kernel $k_1(\boldsymbol{X}, \boldsymbol{Y}) = J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}))^{-1/2} J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{Y}))^{-1/2}$, and the log-Euclidean kernel k_{LE} in the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$. Any rank-one kernel is positive definite, and the log-Euclidean kernel is positive definite for all $\sigma > 0$ according to Section 4.5.2. The Hadamard product of two positive definite kernels is positive definite, thus, k_{WG} is positive definite for all $\sigma > 0$. □

In the next section, we will compare the log-Euclidean kernel presented above with this new wrapped Gaussian kernel on a classification experiment.

4.5.4 Numerical experiments comparing the kernels

Let us now compare numerically the AIRM kernel of Section 4.5.1, the log-Euclidean kernel defined in Equation (4.3) and the wrapped Gaussian kernel defined in Definition 4.16. For this, we performed a similar classification experiment as in Wagner vom Berg et al. (2025): we used a kernel Support Vector Machine (SVM, see Chapter 12 of Hastie et al. 2009) to perform classification.

The datasets We used the following BCI datasets: BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012), BNCI2014004 (Leeb et al. 2007), BNCI2015001 (Faller et al. 2012), Zhou2016 (B. Zhou et al. 2016) from MOABB (Aristimunha, Carrara, et al. 2026), bandpass filtered from $[7, 35]$ Hz. Those datasets contain multiple sessions (5 for BNCI2014004, 3 for Zhou2016 and 2 for the other ones). We used a cross-session validation in a leave-one-session-out scheme for comparing classification performance. We also tested on the two following non-BCI datasets: AirQuality (Smith, Hua, et al. 2022) and Textile (Bergmann et al. 2021) described in Section 4.3.5.

The methodology For all the datasets, the hyperparameters σ of the kernel and C of the SVM were optimized using a grid search in a nested five-fold cross-validation on the training set. The range for those parameters were both $(10^i)_{i=-3,\dots,3}$ as in Wagner vom Berg (2025). We compared a log-Euclidean kernel at the identity (so exactly the Equation (4.3)), a log-Euclidean kernel at the Fréchet mean of the training dataset (see Remark 4.15), a wrapped

Dataset	AIRM	Log-Euclidean Kernel		Wrapped Gaussian Kernel	
		Identity	Fréchet mean	Identity	Fréchet mean
BNCI2014001	**74.0**$_{\pm 13.1}$	72.4$_{\pm 13.7}$	72.5$_{\pm 13.7}$	66.7$_{\pm 12.1}$	68.5$_{\pm 14.3}$
BNCI2014004	69.8$_{\pm 10.6}$	69.6$_{\pm 10.0}$	**69.9**$_{\pm 10.5}$	69.5$_{\pm 10.3}$	69.5$_{\pm 10.4}$
BNCI2015001	75.6$_{\pm 15.2}$	**78.7**$_{\pm 13.4}$	78.0$_{\pm 13.4}$	52.7$_{\pm 4.0}$	74.4$_{\pm 14.0}$
Zhou2016	73.8$_{\pm 9.9}$	78.6$_{\pm 8.8}$	**79.5**$_{\pm 10.8}$	50.5$_{\pm 1.1}$	73.8$_{\pm 9.9}$
Air Quality	96.0$_{\pm 3.7}$	95.1$_{\pm 3.2}$	96.0$_{\pm 3.7}$	92.1$_{\pm 2.4}$	**97.0**$_{\pm 2.4}$
Textile	**94.4**$_{\pm 0.2}$	93.9$_{\pm 0.3}$	94.3$_{\pm 0.2}$	89.0$_{\pm 0.5}$	**94.4**$_{\pm 0.2}$

Table 4.4. Classification accuracy (%, mean ± standard deviation) obtained with the different Riemannian kernels used in an SVM. The best result for each dataset is highlighted in bold. The Fréchet mean is the Fréchet mean of the whole training dataset.

Gaussian kernel where the anchor is the identity and a wrapped Gaussian kernel where the anchor is the Fréchet mean of the training dataset. For the AIRM kernel, it does not depend on any anchor point. However, for this kernel, we made sure that it was positive definite for each σ before using the kernel in the SVM. In practice, we observed that when $\sigma = 10$ or $\sigma = 100$, some AIRM kernels were not positive definite, the other σ's did not cause any issue.

The results The results are given in Table 4.4. There is no clear winner between the different kernels studied. We can see that using a wrapped Gaussian kernel does not help improve the performances of the kernel SVM. Rather, the performances of the log-Euclidean kernel at the Fréchet mean and the wrapped Gaussian kernel also at the Fréchet mean are comparable. Taking into account the distortion induced by the mapping to the tangent space can be mathematically appealing, but does not change real classification accuracies. Nevertheless, one can observe something interesting: the wrapped Gaussian kernel, when anchored at the identity, performs the worst. This is where the effect of the Jacobian determinant can be observed: if the anchor point of the wrapped Gaussian kernel is not wisely chosen, it deteriorates the result of the downstream task (here classification). Thus, choosing the Fréchet mean as the anchor point of the wrapped Gaussian kernel improves the classification result. Moreover, the differences between an intrinsic kernel (the AIRM one) and extrinsic ones (the log-Euclidean or the wrapped Gaussian kernel) are quite marginal, on some datasets the AIRM one performs better, while it is the contrary on other datasets.

4.6 Conclusion

In this chapter, we showed how different tools used for learning on the manifold $\mathcal{P}_d$ of SPD matrices can be unified under a common probabilistic framework based on the isotropic Gaussian and the wrapped Gaussian. We reviewed several tools, from classifiers (MDM, Tangent-Space LDA/QDA) to outlier detection tools (Riemannian potato) and Riemannian Gaussian kernels. This framework is summarized at Figure 4.5 that groups them according to the Gaussian used underneath. Most of the already existing tools relied on the isotropic Gaussian, and we were able to extend them by proposing new tools based on the wrapped Gaussian.

The main advantage of using the wrapped Gaussian is that this distribution is anisotropic by nature, thus it might be able to better model complex data than an isotropic distribution. We thus proposed two new classifiers that we call *Wrapped Discriminant Analysis*: the Ho-WDA and the He-WDA. We showed numerically that, if one had enough data, these new classifiers were able to perform better than the MDM and Tangent-Space LDA/QDA classifiers.

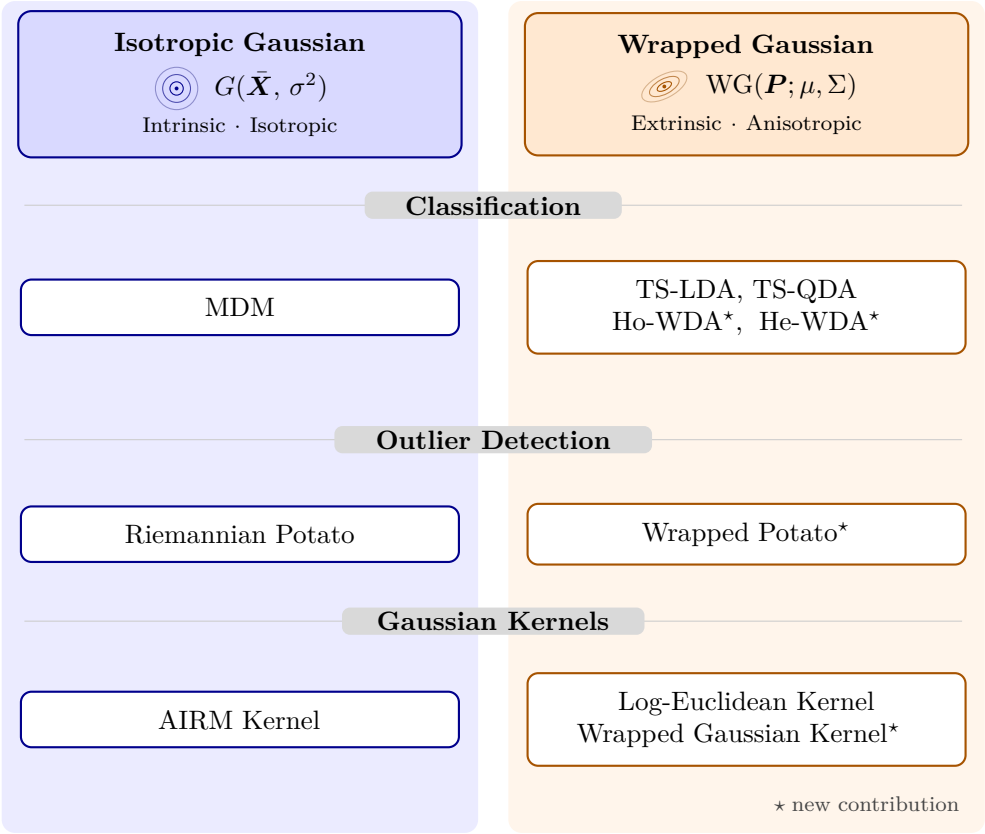


Figure 4.5. Overview of the probabilistic framework developed in this chapter. Each ML tool on $\mathcal{P}_d$ is reinterpreted using either the isotropic or the wrapped Gaussian. Entries marked ⋆ are new contributions of this thesis.

The main difference with these new algorithms is that each class has its own tangent space, leading to a better approximation of the data. Indeed, the anchor of the wrapped Gaussian is now a full parameter, that can be shared by the different classes in the Tangent-Space LDA/QDA, but can also be class specific in the Ho-WDA and He-WDA. Using the wrapped Gaussian model allows us to take into account the deformation induced by the mapping to the tangent space. Taking into account this deformation led us to propose a new Gaussian kernel, the *Wrapped Gaussian Kernel*. In the small experiment we ran, this new kernel did not prove to outperform a classical log-Euclidean Gaussian kernel, where the deformation encoded by the Jacobian determinant of the exponential mapping is not taken into account.

The different tools studied in this chapter give us an answer to question 2. Indeed, using the wrapped Gaussian, we were able to build new learning tools for the manifold $\mathcal{P}_d$. This probabilistic framework will be at the heart of the rest of this manuscript. Indeed, in the next Chapter 5, we will show how this probabilistic framework also applies to Riemannian Flow Matching (Lipman, R. T. Q. Chen, et al. 2023; R. T. Q. Chen and Lipman 2024), a generative model used to sample points from an unknown distribution. We will use the wrapped Gaussian to rewrite already known Flow Matching models, and, using the anchor as a new parameter, we will be able to propose a new Flow Matching algorithm. Later, in Chapter 6, we will study in more depth the *Wrapped Potato* that we started to introduce in Remark 4.11, and we will show that it can be better suited in a real-life scenario of detecting anesthetic states. Finally, in Chapter 7, we will introduce a Riemannian version of the t-SNE (Maaten and Hinton 2008) for the manifold $\mathcal{P}_d$ that will rely on the isotropic Gaussian.

5

Chapter 5

Wrapped Flow Matching on $\mathcal{P}_d$

Contents

5.1 Introduction . . . 111
5.2 Background on Conditional Flow Matching . . . 114
5.2.1 The classical Euclidean Conditional Flow Matching . . . 114
5.2.2 Riemannian Conditional Flow Matching . . . 116
5.3 A global framework unifying CFM on $\mathcal{P}_d$ . . . 117
5.3.1 The general framework . . . 117
5.3.2 Specific parameterizations . . . 119
5.4 Wrapped Conditional Flow Matching . . . 120
5.5 Synthetic experiments . . . 122
5.5.1 The setup . . . 122
5.5.2 The results . . . 123
5.6 Real-world data experiments . . . 125
5.6.1 The setup . . . 125
5.6.2 The results . . . 125
5.7 Conclusion . . . 126

5.1 Introduction

By this point, it is clear that the Gaussian distribution is widely used to model data, and that its Riemannian counterpart studied in this work, the wrapped Gaussian, is able to play this role when the data intrinsically lies on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. We saw in Chapter 4 that using this distribution as a prior on the data could lead to interesting algorithms. However, assuming the Gaussianity of the data (or the wrapped Gaussianity in the case of SPD matrices) is a very strong assumption. It is unlikely that realistic datasets, composed for example of realistic images or covariance matrices estimated from real-life EEGs, satisfy such an assumption. These complex, real-life distributions cannot simply be reduced to Gaussians or wrapped Gaussians and can even be non-parametric distributions. Oftentimes, we do not even know the true underlying distribution: we simply have to rely on samples (sometimes a significant amount) from this distribution. This is exactly the setup we encounter when working with BCIs: we have no idea of the underlying distribution of covariance matrices estimated from the EEGs of a specific subject, and we simply have samples from it.

This setup makes it impossible to use the nice properties of the Gaussian or of the wrapped Gaussian in our scenario. One of the nicest properties of the (wrapped) Gaussian is its ease of sampling. Sampling from the Euclidean Gaussian is a highly studied topic (Thomas et al. 2007) that dates back to the end of the 1950s, starting with the *inverse method* of M. E. Muller (1958) for sampling $\mathcal{N}(0, 1)$, followed by the highly celebrated *Box-Muller algorithm* from Box and M. E. Muller (1958) based on trigonometric transformations. Nowadays, NumPy (Harris et al. 2020) uses the *Ziggurat algorithm*, introduced in Marsaglia and Tsang (2000), to sample from a Gaussian distribution. This method is a rejection-sampling technique based on precomputed rectangular strata under the density curve and is both reliable and very efficient. For the wrapped Gaussian on the manifold of SPD matrices $\mathcal{P}_d$, Algorithm 1 gives us a direct procedure for sampling the wrapped Gaussian, as we have a closed-form expression for the mapping $\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}$.

Flow-based generative models

In order to sample points from an unknown data distribution from which we only have access to samples, recent and popular approaches leverage the Gaussian distribution and transform Gaussian samples into samples from the data distribution. In this chapter, we focus on one of these generative models: *Conditional Flow Matching* (Lipman, R. T. Q. Chen, et al. 2023; X. Liu et al. 2023).

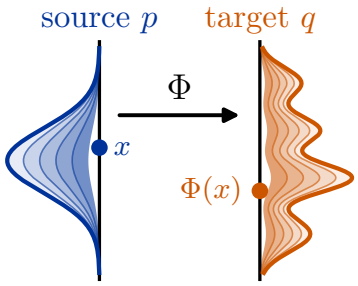


Figure 5.1. Illustration of flow-based generative models where the goal is to learn a transformation Φ that maps samples from p to samples from q. Inspired by https://memorization-generalization.github.io/.

Let p denote a simple source distribution, such as a Gaussian on $\mathbb{R}^d$ or a wrapped Gaussian on $\mathcal{P}_d$. By simple, we mean that the distribution has a simple sampling procedure. Let q denote the unknown target data distribution, from which we only have access to a finite amount of samples. Our goal here is to sample new points from q. The idea behind flow-based generative models is to learn a transformation Φ that will map samples from the source distribution p to samples from the target distribution q. Thus, we want to learn Φ such that the push-forward of p by Φ is equal to q: $\Phi \# p = q$. This is illustrated in Figure 5.1. In Flow Matching (FM) specifically (Lipman, R. T. Q. Chen, et al. 2023; X. Liu et al. 2023; Michael S Albergo and Vanden-Eijnden 2022), this transformation Φ is built from a time-dependent flow Φ_t that solves an Ordinary Differential Equation (ODE). Then, instead of directly learning the transformation Φ, we aim at learning the vector field u_t defining Φ_t using a neural network, and when sampling new points, we follow this vector field u_t by solving the ODE. This flow Φ_t will define probability paths p_t that dictates the distribution of the samples at time t during the sampling process. The difficulty is then finding the correct vector field u_t to learn, based only on samples of the target distribution q. This is where *Conditional Flow Matching* (CFM, Lipman, R. T. Q. Chen, et al. 2023) comes in handy, as it provides a way of *conditioning* the vector field u_t on the known samples, while still recovering the full vector field after learning. This framework has shown state-of-the-art performance in the generation of images (Esser et al. 2024), videos (A. Polyak et al. 2024), speech (Le et al. 2023) or audio (Vyas et al. 2023). It also shares common ground with diffusion models (Ho et al. 2020; Yang Song et al. 2021), with the difference that diffusion models are driven by a Stochastic Differential Equation (Gao et al. 2024).

Enlarging the probabilistic framework

The initial version of CFM was built for Euclidean data in $\mathbb{R}^d$, and the source distribution p was a Euclidean multivariate Gaussian $\mathcal{N}(0, I_d)$. A first generalization of CFM to Riemannian manifolds was proposed by R. T. Q. Chen and Lipman (2024). They generalize the linear interpolation used to build conditional probability paths to geodesic interpolation, yielding Riemannian conditional probability paths. This RiemannianCFM with geodesic interpolation has been used in Bose et al. (2024) to generate novel protein structures on the special Euclidean group SE(3) or in Braun et al. (2024) to learn and generate robot sensorimotor policies on the hypersphere or on SO(3). In Zaghen et al. (2025), the authors developed a Riemannian version of Variational Flow Matching, based on the isotropic Gaussian of Said, Bombrun, et al. (2017). Another approach to adapting CFM to Riemannian manifolds is to use pullback geometries, as in Collas, Ju, et al. (2026). Their so-called *DiffeoCFM* sends the data to a Euclidean space E (for example a specific tangent space with the risk of deforming the distribution) *via* a global diffeomorphism and relies on Euclidean CFM in E before sending back the generated samples to the manifold. In their paper, the authors apply their DiffeoCFM to fMRI and EEG datasets. A similar approach is taken by Z. Zhou et al. (2025), in which the authors use the Cholesky decomposition to perform efficient FM on the simpler manifold of lower-triangular matrices with positive diagonal.

We can start to see some similarities between the Riemannian CFM adaptations described above and the different algorithms described in Chapter 4. For example, the DiffeoCFM is analogous to the Tangent-Space LDA described in Section 4.3.2, as it relies on a single tangent space to project the data and applies the classical Euclidean algorithm in the tangent space. This corresponds exactly to an *extrinsic method* as described in Section 2.3.6. In this chapter, we will extend the probabilistic framework we began building in Chapter 4 to flow matching on $\mathcal{P}_d$. More precisely, we propose two main contributions:

1. **First**, the key novelty is a general framework for CFM on $\mathcal{P}_d$, encompassing state-of-the-art Riemannian CFM methods: *RiemannianCFM* from R. T. Q. Chen and Lipman (2024) and *DiffeoCFM* from Collas, Ju, et al. (2026). For this, we leverage an infinitely supported distribution defined on the manifold $\mathcal{P}_d$, the *wrapped Gaussian* from Chapter 3, to design the conditional probability paths.
2. **Second**, using the flexibility of our framework, we propose a novel interpolation scheme that extends beyond restricted geodesic paths. We call this method *WrappedCFM*.

Outline of the chapter

This chapter is organized as follows: we start by recalling the classical theory of Euclidean CFM and review two state-of-the-art Riemannian CFM methods in Section 5.2. We then introduce our unified framework in Section 5.3 and propose our new interpolation *WrappedCFM* in Section 5.4. Finally, we validate our method on synthetic data in Section 5.5 and on real-world data in Section 5.6.

This work was done in collaboration with a PhD student from IRISA, Aimi Okabayashi and is published in the journal *Transactions on Machine Learning Research*:

> Thibault de Surrel, Aimi Okabayashi, and Nicolas Courty (2026). "Wrapped Flow Matching for Symmetric Positive Definite Matrices". In: *Transactions on Machine Learning Research*

∗ These two authors contributed equally to this work.

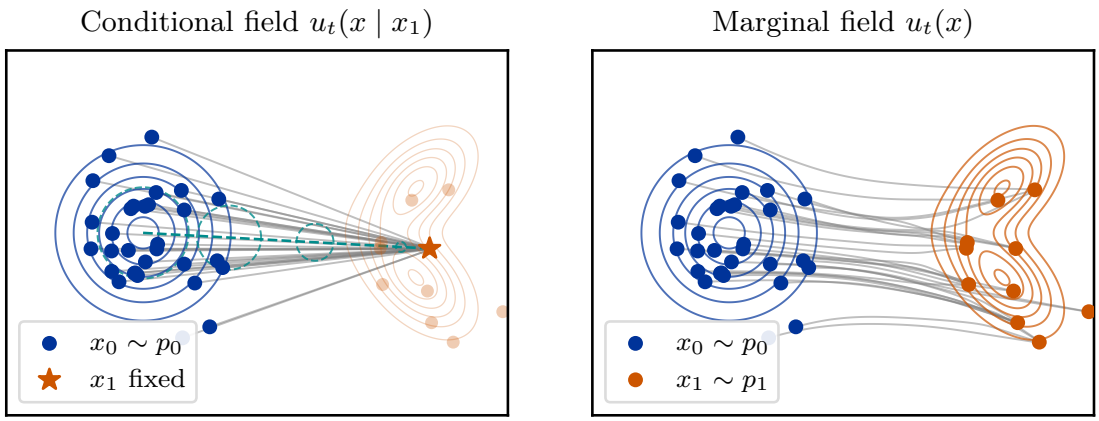


Figure 5.2. Illustration of the Euclidean Conditional Flow Matching. The source distribution p is blue and the target distribution q is in orange. On the left plot, the conditional distribution for a fixed x_1 is shown: the source distribution converges to a Dirac distribution at x_1. On the right plot, the marginal field u_t is shown, once all the conditional fields have been marginalized. Then, the source points flow to different target points.

5.2 Background on Conditional Flow Matching

In this section, we give a brief introduction to the classical theory of Conditional Flow Matching (CFM). The general goal of CFM is to sample points from an unknown data distribution q. In order to do this, CFM learns a parametric map Φ that pushes forward a "simple" distribution p (e.g. a Gaussian) to the data distribution q. We start by describing the classical Euclidean CFM, and then we describe two state-of-the-art approaches to adapt CFM to a Riemannian manifold. For a comprehensive introduction to Flow Matching and CFM, we refer to Lipman, Havasi, et al. (2024).

5.2.1 The classical Euclidean Conditional Flow Matching

Let us first start by describing the classical Euclidean CFM framework on $\mathbb{R}^d$ from Lipman, R. T. Q. Chen, et al. (2023). Let $p = \mathcal{N}(0, I_d)$ be the source distribution and q be the target distribution. In order to build a model capable of generating new samples from q, the Flow Matching framework builds a *probability path* $(p_t)_{0 \leqslant t \leqslant 1}$ such that $p_0 = p$ and $p_1 = q$. To model it, a time dependent flow $\Phi : [0,1] \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ is defined as the solution of the ODE driven by a vector field u_t at time t and initial condition $\Phi_0(x) = x$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(x) = u_t(\Phi_t(x))\,, \quad \Phi_0(x) = x. \tag{5.1}$$

The flow Φ is used to define p_t as $p_t = \Phi_t \# p$, the push-forward of p. Thus, if the flow Φ is correctly built such that $p_1 = \Phi_1 \# p = q$, transforming samples of p with Φ_1 will give a sample of q. Moreover, p_t and u_t are linked by the continuity equation:

$$\partial_t p_t(x) + \operatorname{div}(p_t(x)\, u_t(x)) = 0, \tag{5.2}$$

where div denotes the divergence operator, which assigns to each point the infinitesimal rate of volume expansion induced by the vector field.

Assuming that we know how to define p_t and u_t, the problem of approximating the vector field u_t can directly be cast as a regression problem using a neural network v_t^θ:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\substack{t \sim \mathcal{U}(0,1) \\ x \sim p_t}} \| v_t^\theta(x) - u_t(x) \|^2.$$

The main challenge lies in constructing such a flow without access to the explicit form of q, relying solely on available samples. Indeed, we know where the flow Φ should start (at p), but we do not know where it should end. Therefore, Lipman, R. T. Q. Chen, et al. (2023) introduce CFM by designing p_t as a mixture of tractable conditional probability paths $p_t(\cdot \mid z)$ that vary with some conditioning variable $z \in \mathcal{Z}$:

$$p_t(x) = \int_{\mathcal{Z}} p_t\left(x \mid z\right) \rho\left(z\right) \mathrm{d}z,$$

where $\rho(z)$ is a distribution over the conditioning variable. The choice of the conditional probability path $p_t(\cdot \mid z)$ defines a conditional vector field $u_t(\cdot \mid z)$, such that the marginalization against z of $u_t(\cdot \mid z)$ generates p_t:

$$u_t(x) = \int_{\mathcal{Z}} u_t\left(x \mid z\right) \frac{p_t\left(x \mid z\right) \rho\left(z\right)}{p_t(x)} \mathrm{d}z.$$

Therefore, rather than regressing $u_t(x)$, we regress the conditional vector field $u_t(x \mid z)$:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{\substack{t\sim\mathcal{U}(0,1) \\ z\sim\rho \\ x\sim p_t(\cdot|z)}} \left\| v_t^\theta(x) - u_t(x \mid z) \right\|^2.$$

The power of the CFM framework lies in the fact that $\mathcal{L}_{\mathrm{CFM}}$ and $\mathcal{L}_{\mathrm{FM}}$ are **equal up to a constant** (see Theorem 2 of Lipman, R. T. Q. Chen, et al. 2023). Therefore, their gradients are equal and minimizing $\mathcal{L}_{\mathrm{CFM}}$ will also minimize $\mathcal{L}_{\mathrm{FM}}$.

There are two classical ways of choosing the shape of the conditional probability paths $p_t(\cdot \mid z)$: one relying on a linear interpolation and one relying on a Gaussian interpolation.

- **For the linear interpolation** (X. Liu et al. 2023), the conditioning variable z is a pair $(x_0, x_1) \sim \rho = p \otimes q$, thus a pair of a source point x_0 and a target data point x_1. Then, the conditional probability paths are chosen to be:

  $$p_t(\cdot \mid x_0, x_1) = \delta_{(1-t)x_0 + t x_1},$$

  where δ_y is the Dirac distribution at $y \in \mathbb{R}^d$. This is simply the linear interpolation between the source point x_0 and the target data point x_1. The corresponding conditional flow and vector field can be found in Lipman, Havasi, et al. (2024).
- **For the Gaussian interpolation** (Lipman, R. T. Q. Chen, et al. 2023), the conditioning variable z is simply a target data point $x_1 \sim q$ (in practice, x_1 is a random training point). Then, the conditional probability paths are Gaussians:

  $$p_t(\cdot \mid x_1) = \mathcal{N}(\mu_t(x_1), \sigma_t^2(x_1) I_d).$$

  Typical examples for the maps μ_t and σ_t are:

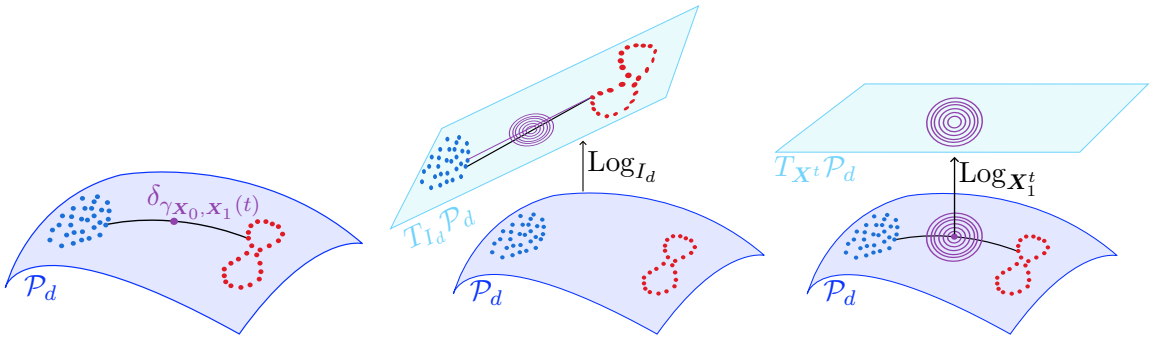
  $$\mu_t(x_1) = t x_1 \text{ and } \sigma_t(x_1) = 1 - (1 - \sigma_{\min})t.$$

  The conditional probability path is then simply an interpolation of Gaussians from $\mathcal{N}(0, I_d)$ at $t = 0$ to $\mathcal{N}(x_1, \sigma_{\min}^2 I_d)$ at $t = 1$, a concentrated Gaussian distribution centered at x_1. Then, the flow pushing p to p_t is

  $$\Phi_t(x) = \sigma_t(x_1) x + \mu_t(x_1) = (1 - (1 - \sigma_{\min})t)x + t x_1, \tag{5.3}$$

  and the corresponding Euclidean conditional vector field $u_t\left(x \mid x_1\right)$ is given by (see Equation 21 of Lipman, R. T. Q. Chen, et al. 2023):

  $$u_t\left(x \mid x_1\right) = \frac{\sigma_t'\left(x_1\right)}{\sigma_t\left(x_1\right)} \left(x - \mu_t\left(x_1\right)\right) + \mu_t'\left(x_1\right) = \frac{x_1 - (1 - \sigma_{\min})x}{1 - (1 - \sigma_{\min})t}. \tag{5.4}$$



(a) RiemannianCFM. **(b)** DiffeoCFM. **(c)** WrappedCFM.

Figure 5.3. Illustration of the different CFM models on the manifold $\mathcal{P}_d$: RiemannianCFM that uses the geodesic interpolation between $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$, DiffeoCFM that relies on a mapping to the tangent space at identity I_d and then uses the Euclidean CFM, and finally, the proposed WrappedCFM that uses a wrapped Gaussian whose anchor moves along the geodesic between I_d and $\boldsymbol{X}_1$.

Remark 5.1. In this case of Gaussian interpolation, Lipman, R. T. Q. Chen, et al. (2023) have shown (see their Example II of Section 4.1), that the conditional flow Φ_t is the *optimal transport displacement map* between the two Gaussians $p_0(\cdot \mid x_1) = \mathcal{N}(0, I_d)$ and $p_1(\cdot \mid x_1) = \mathcal{N}(x_1, \sigma_{\min}^2 I_d)$. This optimal transport displacement map moves a source probability distribution to a target distribution with minimum cost, and the specific map for Euclidean Gaussians is given in Example 1.7 of McCann (1997).

5.2.2 Riemannian Conditional Flow Matching

Now that we have explained how Euclidean CFM works, let us describe two possible ways of adapting it when the data distribution q lives on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. We start by describing *RiemannianCFM* introduced in R. T. Q. Chen and Lipman (2024) that relies on geodesic interpolation and then, we review *DiffeoCFM*, from Collas, Ju, et al. (2026) that leverages the tangent spaces. These two CFM models are geometry-aware and are illustrated in Figure D.4a and Figure D.4b.

5.2.2.1 RiemannianCFM

In R. T. Q. Chen and Lipman (2024), the authors propose to adapt the Euclidean CFM with linear interpolation described in Section 5.2.1 to the case of a Riemannian manifold. For this, the conditioning variable z is a pair $(\boldsymbol{X}_0, \boldsymbol{X}_1) \sim \rho = p \otimes q$ sampled from the source distribution p and the target distribution q (in practice, $\boldsymbol{X}_1$ is uniformly chosen from the training dataset). Then, they leverage geodesics to build the conditional probability paths $p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$, as they are the shortest paths between two points on a manifold. The resulting conditional probability path $p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$ and its associated flow are:

$$\begin{aligned} p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1) &= \delta_{\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)} \\ \Phi_t^{\mathrm{RCFM}}(\boldsymbol{X} \mid \boldsymbol{X}_0, \boldsymbol{X}_1) &= \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t), \end{aligned} \tag{5.5}$$

where $\delta_{\boldsymbol{X}}$ is the Dirac distribution at $\boldsymbol{X}$ and $\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}$ is the unique geodesic from $\boldsymbol{X}_0$ to $\boldsymbol{X}_1$. Then, the conditional vector field $u_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$ that generates $p_t^{\mathrm{RCFM}}(\cdot \mid \boldsymbol{X}_0, \boldsymbol{X}_1)$ is simply

$$\frac{\mathrm{d}}{\mathrm{d}t}\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t). \tag{5.6}$$

Remark 5.2. For RiemannianCFM, the source distribution p can be any distribution. In the rest of this work, we will assume that the source distribution p is always a wrapped

Gaussian $\mathrm{WG}(I_d; 0_D, \sigma_{\text{init}}^2 I_D)$, with $\sigma_{\text{init}} > 0$.

5.2.2.2 DiffeoCFM

The authors of Collas, Ju, et al. (2026) use a Euclidean space E to pullback the distributions p and q onto E and then rely on classical Euclidean CFM in E as described in Section 5.2.1. When the manifold of interest is $\mathcal{P}_d$, the chosen Euclidean space E is the tangent space at identity $T_{I_d}\mathcal{P}_d$ and the pullback is the matrix logarithm $\log = \mathrm{Log}_{I_d}$. Thus, by denoting p_t^{E} the Euclidean conditional probability path and by Φ_t^{E} the Euclidean flow (as given in Lipman, R. T. Q. Chen, et al. 2023), we can express them for *DiffeoCFM*:

$$\begin{aligned} p_t^{\text{DiffeoCFM}}(\cdot \mid \boldsymbol{X}_1) &= \exp \# p_t^{\mathrm{E}}(\cdot \mid \log \boldsymbol{X}_1) \\ \Phi_t^{\text{DiffeoCFM}}(\cdot \mid \boldsymbol{X}_1) &= \exp \circ \Phi_t^{\mathrm{E}}(\cdot \mid \log \boldsymbol{X}_1) \end{aligned} \tag{5.7}$$

In this case, the conditional variable z is the log of a point $\boldsymbol{X}_1 \sim q$. One can also express the conditional vector field generating $p_t^{\text{DiffeoCFM}}(\cdot \mid \boldsymbol{X}_1)$ (see Equation (7) of Collas, Ju, et al. 2026):

$$u_t^{\text{DiffeoCFM}}(\boldsymbol{X} \mid \boldsymbol{X}_1) = \mathrm{d}\exp(\log \boldsymbol{X}) \cdot \left[u_t^{\mathrm{E}}(\log \boldsymbol{X}) \right], \tag{5.8}$$

where $\mathrm{d}\exp(\boldsymbol{U}) \cdot \boldsymbol{V}$ is the differential of exp at $\boldsymbol{U}$ applied at $\boldsymbol{V}$.

5.3 A global framework unifying CFM on $\mathcal{P}_d$

Now that we have described the general approach of CFM and two specific Riemannian methods, *RiemannianCFM* and *DiffeoCFM*, we propose a general CFM framework on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. This framework encompasses *RiemannianCFM* and *DiffeoCFM* but can also propose new Riemannian CFM methods. This framework is partially inspired by the one provided in A. Tong et al. (2024).

5.3.1 The general framework

This framework relies on the wrapped Gaussian distribution described in Chapter 3. In this framework, the source distribution p will be a simple centered wrapped Gaussian: $p = \mathrm{WG}(I_d; 0_D, \sigma_{\text{init}}^2 I_D)$, where we recall that $D = d(d+1)/2$ is the dimension of the tangent spaces of $\mathcal{P}_d$. This distribution is convenient due to its ease of sampling[1] (see Section 3.2). Let us now consider a conditioning variable $\boldsymbol{Z} \in \mathcal{Z}$, such that $\boldsymbol{Z} \sim \rho$, where $\rho(\boldsymbol{Z})$ is a distribution over the conditioning variable. We consider the following conditional probability path $(p_t(\cdot \mid \boldsymbol{Z}))_{0 \leqslant t \leqslant 1}$:

$$p_t(\cdot \mid \boldsymbol{Z}) = \mathrm{WG}(\bar{\boldsymbol{X}}_t(\boldsymbol{Z}); \mu_t(\boldsymbol{Z}), \sigma_t(\boldsymbol{Z})^2 I_D), \tag{5.9}$$

where $\bar{\boldsymbol{X}} \colon [0,1] \times \mathcal{Z} \to \mathcal{P}_d$, $\mu \colon [0,1] \times \mathcal{Z} \to \mathbb{R}^D$ and $\sigma \colon [0,1] \times \mathcal{Z} \to \mathbb{R}_{\geqslant 0}$ parametrize the conditional probability path. This conditional probability path $(p_t(\cdot \mid \boldsymbol{X}_1))_t$ is defined directly on the manifold $\mathcal{P}_d$ and relies on an infinitely supported distribution. In order to match the source distribution p at $t = 0$, we impose that the marginal $p_0 = p$. This marginal p_0 can be computed as $p_0 = \int_{\mathcal{Z}} p_0(\cdot \mid \boldsymbol{Z}) \rho(\boldsymbol{Z}) \mathrm{d}\boldsymbol{Z}$, which constrains jointly the conditional path $p_0(\cdot \mid \boldsymbol{Z})$ and the distribution ρ over the conditioning variable $\boldsymbol{Z}$.

Remark 5.3. Using Proposition 3.13, one can see that the spread of $\mathrm{WG}(I_d; 0_D, I_D)$ scales roughly like d. This can lead to some numerical issues when d is large. This is why, to solve

[1] In this specific application, the ease of sampling of the wrapped Gaussian is a major argument against the isotropic Gaussian from Said, Bombrun, et al. (2017) or the entropy-maximizing Gaussian from Pennec (2006) that are described in Section 2.3.4.

this issue, we start from $p_0 = \mathrm{WG}(I_d; 0_D, \sigma_{\text{init}}^2 I_D)$ with $\sigma_{\text{init}} \ll 1$, thus a more concentrated source distribution. In the experiments, we will set $\sigma_{\text{init}} = \sqrt{\frac{2}{d(d+1)}}$.

We now need to build a conditional flow Φ_t (conditioned by $\boldsymbol{Z}$) such that $\Phi_t \# p_0 = p_t(\cdot \mid \boldsymbol{Z})$. Using the properties of the wrapped Gaussian given in Proposition 3.8, we have the following proposition:

Proposition 5.4. *The following conditional flow Φ_t satisfies $\Phi_t \# p_0 = p_t(\cdot \mid \boldsymbol{Z})$ where $p_t(\cdot \mid \boldsymbol{Z})$ is given in Equation (5.9)*

$$\Phi_t(\boldsymbol{X}) = \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \exp\left[\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{init}} \log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right] \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}. \tag{5.10}$$

This conditional flow Φ_t can be decomposed into two parts: the first one is inside the exp, where one retrieves the classical Euclidean flow between two Gaussians (up to the change of variables $\log(\boldsymbol{X})$) as given in Equation (5.3), this part affects the parameters of the Gaussian in the tangent space. Then, the second part is the outside of the exp, that controls and adapts the anchor point of the wrapped Gaussian defining the conditional probability path.

This flow then provides a vector field that generates the conditional probability path $(p_t(\cdot \mid \boldsymbol{Z}))_t$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) = u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}) \in T_{\Phi_t(\boldsymbol{X})}\mathcal{P}_d.$$

One can solve this ODE for u_t to get a closed form of the conditional vector field:

Theorem 5.5. *Let $p_t(\cdot \mid \boldsymbol{Z})$ be the conditional probability path given in Equation (5.9) and Φ_t its flow given in Equation (5.10). The vector field defining Φ_t depends on the value of $\sigma_t(\boldsymbol{Z})$ in the following manner:*

- *If $\sigma_t(\boldsymbol{Z}) > 0$, then the unique vector field defining Φ_t is*

$$\begin{aligned} u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = {} & \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y} + \boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) \\ & + \bar{\boldsymbol{X}}_t^{1/2}\,\mathrm{d}\exp(\log \boldsymbol{M}_t) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{M}_t - \mu_t) + \mu_t'\right]\bar{\boldsymbol{X}}_t^{1/2}, \end{aligned}$$

  where $\boldsymbol{S}$ is the derivative of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}$, $\boldsymbol{M}_t = \bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}$, and all time-dependent quantities are evaluated at $\boldsymbol{Z}$.
- *If $\sigma_t(\boldsymbol{Z}) = 0$, then*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t^{1/2}\exp(\mu_t)\bar{\boldsymbol{X}}_t^{1/2}\right],$$

  where all time-dependent quantities are evaluated at $\boldsymbol{Z}$.

The proof can be found in Appendix A.3.1. Let us comment on this result. First, the distinction between the cases $\sigma_t(\boldsymbol{Z}) = 0$ and $\sigma_t(\boldsymbol{Z}) > 0$ is due to the fact that when $\sigma_t(\boldsymbol{Z}) = 0$, the conditional flow Φ_t defined in Equation (5.10) is not invertible. Second, the time derivative $\boldsymbol{S}$ of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}$ is the solution to the following Sylvester equation (see Theorem 1.26 or Chapter 6 of Higham 2008):

$$\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\,\boldsymbol{S}(t) + \boldsymbol{S}(t)\,\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} = \frac{\mathrm{d}}{\mathrm{d}t}\bar{\boldsymbol{X}}_t(\boldsymbol{Z}), \tag{5.11}$$

which can be numerically solved. However, for the specific choices of $\bar{\boldsymbol{X}}_t(\boldsymbol{Z})$ considered in the remainder of this work, $\boldsymbol{S}$ can always be computed explicitly. Third, the direction of the exp differential is exactly the conditional vector field for the Euclidean CFM as given in Theorem 3 of Lipman, R. T. Q. Chen, et al. (2023). Finally, recall that $u_t(\cdot \mid \boldsymbol{Z})$ is a vector field on $\mathcal{P}_d$ and that $T_{\boldsymbol{X}}\mathcal{P}_d \cong \mathcal{S}_d$. One can verify that $u_t(\boldsymbol{X} \mid \boldsymbol{Z})$ is indeed symmetric.

Method	$\rho(\boldsymbol{Z})$	$\bar{\boldsymbol{X}}_t(\boldsymbol{Z})$	$\mu_t(\boldsymbol{Z})$	$\sigma_t(\boldsymbol{Z})$
RiemannianCFM	$p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$	$\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)$	0_D	0
DiffeoCFM	$q(\boldsymbol{X}_1)$	I_d	$t\log(\boldsymbol{X}_1)$	σ_{lin}
WrappedCFM	$q(\boldsymbol{X}_1)$	$\boldsymbol{X}_1^t$	0_D	σ_{lin}

Table 5.1. Unified Wrapped-Gaussian framework for Riemannian CFM on $\mathcal{P}_d$. Each method corresponds to a specific parameterization of the conditional path $p_t(\cdot \mid \boldsymbol{Z}) = \text{WG}(\bar{\boldsymbol{X}}_t; \mu_t, \sigma_t^2 I_D)$. We denote the linear scheduler $\sigma_{\text{lin}} = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\text{min}})\, t$

5.3.2 Specific parameterizations

The newly built general Riemannian CFM framework for the manifold $\mathcal{P}_d$ relies on the parameterization of the conditional probability paths $(p_t(\cdot \mid \boldsymbol{Z}))_t$ by the three following mappings: $\bar{\boldsymbol{X}}$, μ and σ and by the choice of the distribution over the conditioning variable $\boldsymbol{Z}$. We will now see that, depending on the choices of these parameterizations, we are able to retrieve the SOTA methods described in Section 5.2: the *RiemannianCFM* and the *DiffeoCFM*. We summarize the different parameterizations in Table 5.1.

5.3.2.1 RiemannianCFM

Let us start with the RiemannianCFM. In order to retrieve this method from our general framework, we chose the conditioning variable $\boldsymbol{Z}$ to be a pair of a random source point $\boldsymbol{X}_0 \sim p$ and a random target point $\boldsymbol{X}_1 \sim q$ thus, $\boldsymbol{Z} = (\boldsymbol{X}_0, \boldsymbol{X}_1)$ and $\rho(\boldsymbol{Z}) = p(\boldsymbol{X}_0)q(\boldsymbol{X}_1)$ is the independent coupling. Then, the choice for the parameterization of the conditional probability paths is:

$$\bar{\boldsymbol{X}}_t(\boldsymbol{Z}) = \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t),\ \mu_t(\boldsymbol{Z}) = 0_D,\ \text{and}\ \sigma_t(\boldsymbol{Z}) = 0,$$

where we recall that $\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}$ is the unique geodesic joining $\boldsymbol{X}_0$ and $\boldsymbol{X}_1$. This is the degenerated case where the dispersion of the wrapped Gaussian is zero, leading to a Dirac distribution on $\mathcal{P}_d$. It is similar to the Euclidean setting as described in A. Tong et al. (2024) (in which it is referred to as "Basic form of CFM").

Let us verify that this parameterization does indeed lead to RiemannianCFM: when, $\mu_t(\boldsymbol{Z}) = 0$ and $\sigma_t(\boldsymbol{Z}) = 0$, one can verify that the conditional flow Φ_t given in Equation (5.10) does not depend on the point $\boldsymbol{X}$ anymore:

$$\Phi_t(\boldsymbol{X} \mid \boldsymbol{X}_0, \boldsymbol{X}_1) = \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)^{1/2} \exp(0_D) \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t)^{1/2} = \gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t),$$

which is the conditional vector field given in Equation (5.5). Moreover, using the degenerate case of Theorem 5.5, we have

$$u_t(\boldsymbol{Y} \mid \boldsymbol{X}_0, \boldsymbol{X}_1) = \frac{\mathrm{d}}{\mathrm{d}t}\gamma_{\boldsymbol{X}_0,\boldsymbol{X}_1}(t),$$

which is the conditional vector field of the RiemannianCFM as given in R. T. Q. Chen and Lipman (2024) and at Equation (5.6).

5.3.2.2 DiffeoCFM

In order to get DiffeoCFM from our general framework, the conditioning variable $\boldsymbol{Z}$ is chosen to be a single random point $\boldsymbol{X}_1$ from the target distribution q, and one chooses the following parameterization:

$$\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = I_d,\ \mu_t(\boldsymbol{X}_1) = t\log(\boldsymbol{X}_1),\ \text{and}\ \sigma_t(\boldsymbol{X}_1) = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\text{min}})\, t.$$

In DiffeoCFM, everything is sent to the tangent space $T_{I_d}\mathcal{P}_d$ at identity, so the anchor point $\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1)$ of the WG does not change and then classical CFM is applied in $T_{I_d}\mathcal{P}_d$. Going back to the conditional flow Φ_t as given in Equation (5.10), we have:

$$\Phi_t(\boldsymbol{X}) = \exp\left[\frac{\sigma_t}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + t\log(\boldsymbol{X}_1)\right],$$

which is exactly the flow as given in Equation (5.7) with the Euclidean flow being the one from Equation (5.3). Let us now focus on the conditional vector field given in Theorem 5.5. As $\boldsymbol{X}_t(\boldsymbol{X}_1) = I_d$, the time derivative of $\boldsymbol{X}_t(\boldsymbol{X}_1)^{1/2}$ is 0, thus, $\boldsymbol{S}(t) = 0$. Then, the conditional vector field is:

$$\begin{aligned} u_t(\boldsymbol{Y} \mid \boldsymbol{X}_1) &= \mathrm{d}\exp(\log(\boldsymbol{Y})) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{Y} - t\log \boldsymbol{X}_1) + \log \boldsymbol{X}_1\right] \\ &= \mathrm{d}\exp(\log(\boldsymbol{Y})) \cdot \left[\frac{\sigma_{\text{init}}\log \boldsymbol{X}_1 - (\sigma_{\text{init}} - \sigma_{\min})\log \boldsymbol{Y}}{\sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t}\right], \end{aligned}$$

which is exactly the conditional vector field $u_t^{\text{DiffeoCFM}}$ from Equation (5.8) where the Euclidean conditional vector field is the one from Equation (5.4) (where the starting Gaussian is $\mathcal{N}(0, \sigma_{\text{init}}^2 I_d)$ instead of $\mathcal{N}(0, I_d)$).

5.4 Wrapped Conditional Flow Matching

After introducing a general framework for CFM on the Riemannian manifold $\mathcal{P}_d$ of SPD matrices and showing how state-of-the-art methods can be recovered through specific parameterizations, we introduce in this section a new parameterization that leads to a novel Riemannian CFM model. We call this method *WrappedCFM*. This WrappedCFM relies on a conditional probability path that is denoted $(p_t^{\text{WCFM}})_{0\leqslant t\leqslant 1}$. Like DiffeoCFM, the conditioning variable $\boldsymbol{Z}$ is simply a random point $\boldsymbol{X}_1$ from the target distribution q. Then, $(p_t^{\text{WCFM}})_t$ is built using the following parameterization of the general conditional probability paths given in Equation (5.9):

$$\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = \boldsymbol{X}_1^t,\ \mu_t(\boldsymbol{X}_1) = 0_D,\ \text{and}\ \sigma_t(\boldsymbol{X}_1) = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t, \tag{5.12}$$

where $\sigma_{\min} \ll \sigma_{\text{init}}$. Therefore, the conditional probability path at time $t = 0$ is $p_0^{\text{WCFM}}(\cdot \mid \boldsymbol{X}_1) = \mathrm{WG}(I_d; 0_D, \sigma_{\text{init}}^2 I_D) = p$, and at time $t = 1$, is a concentrated WG around $\boldsymbol{X}_1$. This conditional probability path $(p_t^{\text{WCFM}})_{0\leqslant t\leqslant 1}$ is the equivalent of the classical Gaussian conditional probability paths from Lipman, R. T. Q. Chen, et al. (2023) but defined on $\mathcal{P}_d$ using wrapped Gaussians. Remark that $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{X}_1)$ is the unique geodesic between I_d and $\boldsymbol{X}_1$ under the AIRM metric given in Definition 2.13 (see Proposition 2.14).

The differences between the proposed WrappedCFM and RiemannianCFM are that the WrappedCFM relies on an infinitely supported distribution, whereas the RiemannianCFM relies on Dirac distributions (recall that RiemannianCFM is a degenerate case of our framework, when $\sigma_t(\boldsymbol{X}_1) = 0$). The difference between WrappedCFM and DiffeoCFM is that the DiffeoCFM uses only *one* tangent space to project the data onto. This tangent space $T_{I_d}\mathcal{P}_d$ is a good local approximation of the manifold $\mathcal{P}_d$ around I_d, but if the target distribution q is very spread out, relying only on $T_{I_d}\mathcal{P}_d$ can induce some deformations (Jaquier, Rozo, and Asfour 2024). In our proposed WrappedCFM, as the anchor $\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1)$ of the wrapped Gaussian modeling the conditional probability path $p_t(\cdot \mid \boldsymbol{X}_1)$ changes smoothly at each time step t, it provides an approximation of $\mathcal{P}_d$ that follows the center of mass of $p_t(\cdot \mid \boldsymbol{X}_1)$. The differences are illustrated in Figure 5.3.

We now need to build a conditional flow Φ_t (conditioned by $\boldsymbol{X}_1$) such that $\Phi_t \# p_0^{\text{WCFM}} = p_t^{\text{WCFM}}(\cdot \mid \boldsymbol{X}_1)$. Using the general flow given in Equation (5.10) and the specific parameterization given in Equation (5.12), we have that the flow Φ_t^{WCFM} is:

$$\Phi_t^{\text{WCFM}}(\boldsymbol{X}) = \boldsymbol{X}_1^{t/2} \boldsymbol{X}^{\frac{\sigma_t(\boldsymbol{X}_1)}{\sigma_{\text{init}}}} \boldsymbol{X}_1^{t/2}. \tag{5.13}$$

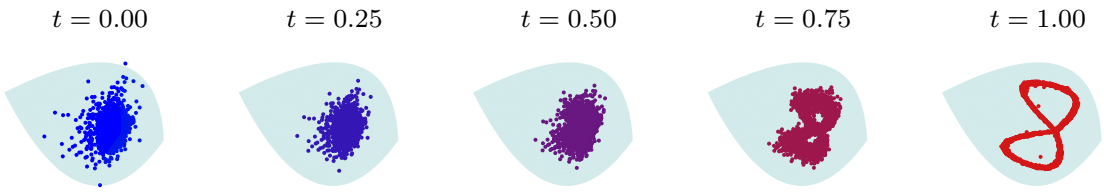


Figure 5.4. Evolution of samples under the Riemannian Euler integration of the learned WrappedCFM vector field, from the initial wrapped Gaussian ($t = 0$) to the target *Figure-8* distribution ($t = 1$). The color encodes the time t, from $t = 0$ (blue) to $t = 1$ (red).

In this case, σ_t is invertible, so one can apply the first case of Theorem 5.5 to get the conditional vector field that generates $p_t^{\mathrm{WCFM}}(\cdot \mid \boldsymbol{X}_1)$:

Theorem 5.6. *Let $p_t^{WCFM}(\cdot \mid \boldsymbol{X}_1)$ be the conditional probability path given by the Wrapped-CFM and Φ_t^{WCFM} its corresponding flow map as in Equation (5.13). Then, the unique vector field that defines Φ_t^{WCFM} has the form:*

$$u_t^{WCFM}(\boldsymbol{Y} \mid \boldsymbol{X}_1) = \tfrac{1}{2}\log(\boldsymbol{X}_1)\,\boldsymbol{Y} + \tfrac{1}{2}\boldsymbol{Y}\log(\boldsymbol{X}_1) + \tfrac{\sigma_t'}{\sigma_t}\,\boldsymbol{X}_1^{t/2}\boldsymbol{M}_t^{1/2}\log(\boldsymbol{M}_t)\boldsymbol{M}_t^{1/2}\boldsymbol{X}_1^{t/2},$$

where $\boldsymbol{M}_t = \boldsymbol{X}_1^{-t/2}\boldsymbol{Y}\boldsymbol{X}_1^{-t/2}$ and all time-dependent quantities are evaluated at $\boldsymbol{X}_1$. Consequently, $u_t(\cdot \mid \boldsymbol{X}_1)$ generates the wrapped Gaussian path $p_t(\cdot \mid \boldsymbol{X}_1)$.

In order to prove this theorem, we first need to recall a lemma.

Lemma 5.7 (Equation 10.3 of Serre (2002))**.** *Let $A \in \mathbb{R}^{d\times d}$, then*

$$\frac{\mathrm{d}}{\mathrm{d}t}\exp(tA) = A\exp(tA) = \exp(tA)A.$$

We can now give the proof of Theorem 5.6.

Proof of Theorem 5.6. Using Lemma 5.7, one gets that

$$\boldsymbol{S}(t) = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{X}_1^{t/2} = \frac{\mathrm{d}}{\mathrm{d}t}\exp\left(\frac{t}{2}\log(\boldsymbol{X}_1)\right) = \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{X}_1^{t/2} = \frac{1}{2}\boldsymbol{X}_1^{t/2}\log(\boldsymbol{X}_1).$$

Moreover, let us consider $\boldsymbol{W} \in \mathcal{P}_d$ and $\boldsymbol{U} \in \mathcal{S}_d$ such that they commute. Then, one has that $\mathrm{d}\exp(\boldsymbol{W})\cdot[\boldsymbol{U}] = \exp(\boldsymbol{W})\boldsymbol{U}$ (see Equation 10.15 of Higham 2008). Using this, we get:

$$\begin{aligned}
u_t(\boldsymbol{Y} \mid \boldsymbol{X}_1) &= \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{X}_1^{t/2}\boldsymbol{X}_1^{-t/2}\boldsymbol{Y} + \frac{1}{2}\boldsymbol{Y}\boldsymbol{X}_1^{-t/2}\boldsymbol{X}_1^{t/2}\log(\boldsymbol{X}_1) \\
&\qquad + \boldsymbol{X}_1^{t/2}\mathrm{d}\exp(\log(\boldsymbol{M}_t))\cdot\left[\frac{\sigma_t'}{\sigma_t}\log\boldsymbol{M}_t\right]\boldsymbol{X}_1^{t/2} \\
&= \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{Y} + \frac{1}{2}\boldsymbol{Y}\log(\boldsymbol{X}_1) + \boldsymbol{X}_1^{t/2}\boldsymbol{M}_t\frac{\sigma_t'}{\sigma_t}\log(\boldsymbol{M}_t)\boldsymbol{X}_1^{t/2} \\
&= \frac{1}{2}\log(\boldsymbol{X}_1)\boldsymbol{Y} + \frac{1}{2}\boldsymbol{Y}\log(\boldsymbol{X}_1) + \boldsymbol{X}_1^{t/2}\boldsymbol{M}_t^{1/2}\frac{\sigma_t'}{\sigma_t}\log(\boldsymbol{M}_t)\boldsymbol{M}_t^{1/2}\boldsymbol{X}_1^{t/2},
\end{aligned}$$

where $\boldsymbol{M}_t = \boldsymbol{X}_1^{-t/2}\boldsymbol{Y}\boldsymbol{X}_1^{-t/2}$. This is exactly the conditional vector field from Theorem 5.6. □

Remark 5.8. As stated after Theorem 5.5, we explicitly know how to compute the derivative $\boldsymbol{S}$ of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = \boldsymbol{X}_1^t$ using Lemma 5.7. Thus, we do not need to solve numerically the Sylvester equation give in Equation (5.11) as in the general case.

Finally, the CFM objective is adapted from R. T. Q. Chen and Lipman (2024) to the Riemannian setting of $\mathcal{P}_d$:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,\boldsymbol{X}_1,\boldsymbol{X}} \| v_t^\theta(\boldsymbol{X}) - u_t^{\text{WCFM}}(\boldsymbol{X} \mid \boldsymbol{X}_1) \|_{\boldsymbol{X}}^2, \tag{5.14}$$

where $t \sim \mathcal{U}(0,1)$, $\boldsymbol{X}_1 \sim q$, $\boldsymbol{X} \sim p_t^{\text{WCFM}}(\cdot \mid \boldsymbol{X}_1)$ and $\| \cdot \|_{\boldsymbol{X}}$ is the norm induced by the affine-invariant metric given in Definition 2.13 in the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$.

Once the vector field u_t is learned using v_t^θ, to generate a sample $\tilde{\boldsymbol{X}}_1 \sim q$, one draws an initial sample $\boldsymbol{X}_0 \sim p_0^{\text{WCFM}} = \text{WG}(I_d; 0_D, \sigma_{\text{init}}^2 I_D)$ and integrates the learned vector field v_t^θ from $t = 0$ to $t = 1$ using a Riemannian Euler scheme:

$$\boldsymbol{X}_{k+1} = \text{Exp}_{\boldsymbol{X}_k}(h v_{t_k}^\theta(\boldsymbol{X}_k)) \tag{5.15}$$

where h is a time step usually set to $\frac{1}{100}$ in the experiments and $t_k = kh$. We give an illustration of the solving of the ODE using this Riemannian Euler scheme once the vector field v_t^θ of the WrappedCFM has been learned in Figure 5.4.

5.5 Synthetic experiments

Now that we have introduced a global framework for the different CFM methods using wrapped Gaussians, and proposed a new method for doing CFM on $\mathcal{P}_d$, namely *WrappedCFM*, we want to compare the different methods numerically[2]. We start with synthetic experiments to compare them. Let us start by detailing the setup of this experiment, including the contestants, the datasets descriptions as well as the metrics used.

5.5.1 The setup

The contestants In the following experiments, we will compare our proposed method, WrappedCFM, with the other Riemannian CFM methods that were introduced in Section 5.2.2: *RiemannianCFM* and *DiffeoCFM*. We will also compare these three Riemannian approaches to a classical Euclidean CFM. In this case, the SPD matrices are treated as Euclidean symmetric matrices by vectorizing their upper-triangular part, and the Riemannian geometry is not taken into account.

All the methods rely on a deep network to learn v_t^θ, the architectures of which are detailed in Appendix B.1.

Datasets In this synthetic experiment, we consider two different datasets that each serve different purposes:

1. We first consider two 2×2 datasets. As the matrices are 2×2 SPD matrices, they are symmetric, thus of the form $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$, and one can plot them in 3D by plotting the tuple (a, b, c). These datasets are inspired by the ones used in Z. Zhou et al. (2025), and we call them *Swiss-Roll* and *Figure-8*. The goal of these two datasets is to see qualitatively if the WrappedCFM produces coherent samples as one can plot them in 3D.
2. The second dataset is composed of 10×10 SPD matrices sampled from a mixture of 3 wrapped Gaussians $\text{WG}(\boldsymbol{X}_k; 0_{55}, \sigma_{\text{init}}^2 I_{55})$ for $k = 1, 2, 3$. The goal of this dataset is to have a simple target, but controlling the condition number κ of the class centers $\boldsymbol{X}_k$. For each experiment, a unique $\kappa \in \{10, 10^2, 10^3, 10^4\}$ is chosen, and these class centers $\boldsymbol{X}_k$ are built to have condition number exactly κ. We want to see how the performance of the different FM methods evolves when the conditioning of the dataset worsens. The condition number of a matrix measures how "stretched" or anisotropic this matrix is

[2]Our code is available at https://github.com/thibaultdesurrel/Wrapped_Flow_Matching_SPD.

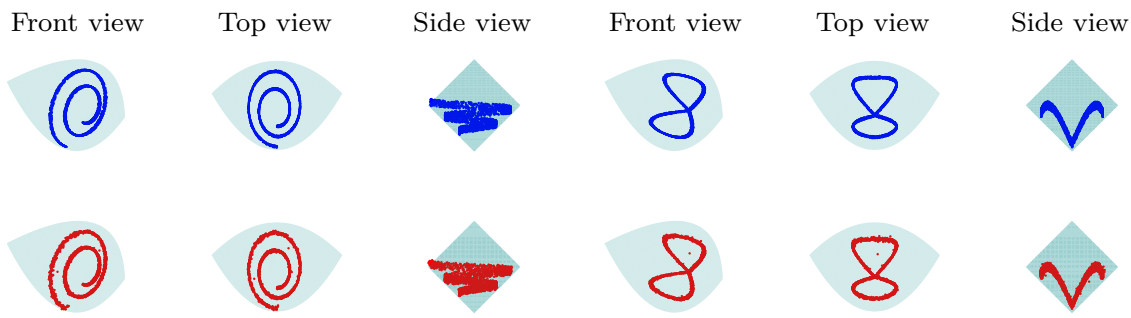


Figure 5.5. Two examples of the result of the proposed WrappedCFM once the flow is learned. The true target distribution is on the top row and the learned one on the bottom row. In these examples (described in Section 5.5), each point represents a 2×2 SPD matrix, that can thus be represented in 3D. We can see that, qualitatively, the proposed WCFM is able to learn the correct distributions.

Dataset	EuclideanCFM	RiemannianCFM	DiffeoCFM	WrappedCFM
Swiss-Roll	$0.375_{\pm 0.273}$	$0.291_{\pm 0.021}$	$0.266_{\pm 0.015}$	$\mathbf{0.265}_{\pm 0.015}$
Figure-8	$0.310_{\pm 0.157}$	$0.219_{\pm 0.021}$	$0.206_{\pm 0.016}$	$\mathbf{0.201}_{\pm 0.020}$

Table 5.2. W_2 distance between the sampled distribution using the different CFM methods and the true distribution for the 2×2 synthetic datasets (Swiss-Roll and Figure-8). Results are reported as mean $\pm$ standard deviation over 5 random seeds. Lower is better. Best results are bold.

(Goldstine and Von Neumann 1951). When its condition number is very large compared to one, the matrix is said to be *ill-conditioned*, and the computation of its inverse, or solution of a linear system of equations is prone to large numerical errors.

More details on how the different datasets are generated are given in Appendix B.1. We ran five times those experiments using five different seeds.

The metrics Let us now detail the different metrics we will use to evaluate the different models. The first metric used in this synthetic experiment will be **the number of sampled matrices that are SPD**. Indeed, our main goal is to build CFM methods that respect the geometry of $\mathcal{P}_d$, so we want the sampled points to be SPD. The three Riemannian methods are built to ensure the SPDness of the sampled matrices, but it is not the case for the Euclidean CFM. Then, the main metric used to evaluate the different CFM methods is the **Wasserstein-2 distance** between the points sampled and the true test distribution. We recall that the Wasserstein-2 distance between the true sample distribution q and the learned one $\tilde{q}$ is defined as follows (Peyré and Cuturi 2019):

$$W_2(q, \tilde{q})^2 = \inf_{\pi \in \Pi(q,\tilde{q})} \mathbb{E}_{(x,y)\sim\pi} \delta^{\text{AIRM}}(x, y)^2,$$

where $\Pi(q, \tilde{q})$ is the set of couplings of q and $\tilde{q}$, and δ^{AIRM} is the AIRM distance given in Equation (2.4). It was computed using POT (Flamary et al. 2024).

5.5.2 The results

We start by giving some qualitative results for the 2×2 datasets *Swiss-Roll* and *Figure-8* by plotting the true and learned distributions in Figure 5.5. We can see that the proposed WrappedCFM method is able to correctly reproduce the true datasets. The learned distributions using our WrappedCFM are noisier than the true distributions, and one can see some outliers. However, the global shape of the two true datasets is well reproduced, which validates our approach. For the 2×2 datasets, we also give the quantitative results of the

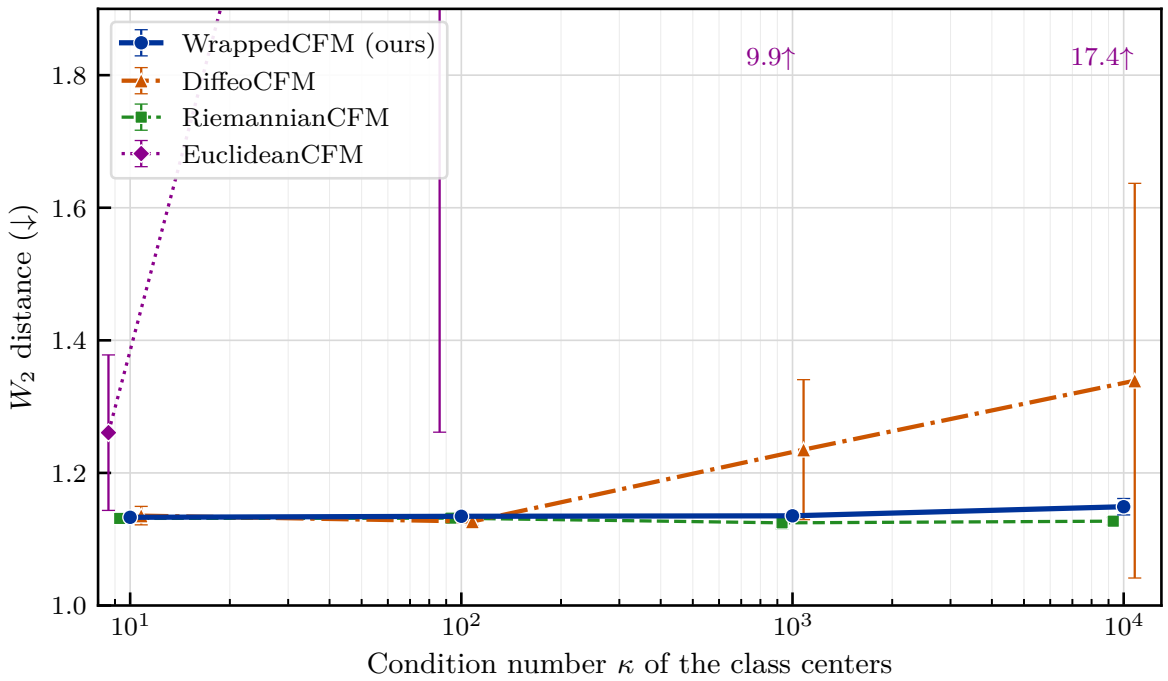


Figure 5.6. W_2-distance (↓, log-log) as a function of the condition number κ of the class centers on the *WG Mixture* dataset. WrappedCFM and RiemannianCFM stay flat and tight across the whole range of κ. DiffeoCFM, by contrast, degrades monotonically as κ grows. EuclideanCFM fares far worse still, collapsing by more than an order of magnitude at $\kappa = 10^4$.

W_2 distance for the different models in Table 5.2. We see that the proposed WrappedCFM achieves the lowest W_2. For the EuclideanCFM, for these datasets, this method was able to produce 99.9% of SPD matrices. The datasets being low-dimensional and well conditioned, this is not surprising.

For the experiment on the dataset composed of mixture of wrapped Gaussians, let us first focus on the percentage of generated samples that are SPD. For the methods taking into account the geometry (RiemannianCFM, DiffeoCFM and WrappedCFM), 100% of the generated samples are SPD. Similarly, for the EuclideanCFM, when $\kappa = 10$ (the conditioning number of the centers of the mixture components), 99.9% of generated samples are SPD. However, when κ grows, the percentage of SPD samples significantly drops: for $\kappa = 10^2$, we have 96.0% of SPD matrices, for $\kappa = 10^3$, only 61.2% are SPD matrices and when $\kappa = 10^4$, only 1.8% of generated samples are SPD. This shows that taking into account the geometry is crucial. When the generated samples were not SPD, we projected them onto $\mathcal{P}_d$ using a convex shrinkage toward the identity, calibrated so that the smallest eigenvalue after shrinkage is exactly 10^{-8} (see Section 2.1.4.2 or Ledoit and Wolf 2004). Then, we computed the W_2 distance to the true distribution. The results are given in Figure 5.6. We can see that the W_2 distance for the EuclideanCFM explodes when κ grows, showing that projecting non-SPD samples on the cone $\mathcal{P}_d$ is not a good solution. For small conditioning number κ, the geometry-aware methods behave all the same, while when κ grows, one can see that DiffeoCFM starts to perform worse than RiemannianCFM and WrappedCFM with a +18% W_2 from $\kappa = 10$ to $\kappa = 10^4$, together with a larger seed-to-seed variance. This can be explained because its fixed-tangent-chart interpolation grows increasingly anisotropic.

We also report training and sampling times in Appendix B.2.

Dataset	Application	Dim.	# Matrices (train/test)	# Classes	Reference
BNCI2014002	BCI	15×15	1,587/177	2	(Steyrl et al. 2016)
BNCI2015001	BCI	13×13	3888/432	2	(Faller et al. 2012)
Textile	Image Analysis	10×10	14,400/1,600	2	(Bergmann et al. 2021)
BreizhCrops	Multispectral Images	13×13	327,760/80,421	4	(Rußwurm et al. 2020)

Table 5.3. Summary of datasets used in the experiments.

5.6 Real-world data experiments

In this section, we will now compare the different Riemannian CFM methods on four different real datasets. We will assess both the quality of the synthetic distribution, and also compare their performance using a classification downstream task.

5.6.1 The setup

The datasets We evaluate our proposed WrappedCFM framework on 4 real-world datasets from different downstream applications: BNCI2014002 (Steyrl et al. 2016) and BNCI2015001 (Faller et al. 2012) from Brain-Computer Interfaces (BCI), Textile (Bergmann et al. 2021) for anomaly detection for textile manufacturing, and BreizhCrops (Rußwurm et al. 2020), a satellite image time series classification dataset. All these datasets include time series from which we computed covariance matrices. Details on preprocessing are presented in Appendix B.3.1 and the datasets are summarized in Table 5.3.

Evaluation procedure After training, each model generates a set of synthetic SPD matrices conditioned on the class labels of both the training and test splits. We measure the quality of the synthetic distribution through two complementary lenses.

1. First, we measure **distribution quality metrics**: *Density*, *Coverage* from Naeem et al. (2020) and *Wasserstein-2 distance* between real and synthetic data. The *Density* evaluates the fidelity of the data, that is, how realistic the synthetic data is while the *Coverage* evaluates the diversity of the data, to see how well the synthetic data captures the variations from the real data. According to Naeem et al. (2020), these metrics improve over the classical *Precision/Recall* from Kynkäänniemi et al. (2019). We use the code provided by Naeem et al. (2020) to compute these metrics. These are reported for "Train vs. Synthetic Train", "Test vs. Synthetic Test", and we also provide "Train vs. Test" metrics on real data for comparison.
2. Second, we assess **downstream utility** via a classifier. The classifier is a logistic regression classifier in the tangent space at the Fréchet mean, under three regimes:
   (i) **T**rained on **R**eal data and **T**ested on **R**eal data (**TRTR**), serving as the upper bound and a control, captures the intrinsic task difficulty.
   (ii) **T**rained on **S**ynthetic data and **T**ested on **R**eal data (**TSTR**). It measures to what point can synthetic data replace real data for learning.
   (iii) **T**rained on **R**eal data and **T**ested on **S**ynthetic data (**TRTS**). The question asked here is: are synthetic samples recognizable as their conditioning class?

To ensure a fair comparison with the competing methods, we did not rely on a wrapped classifier but instead use the same evaluation pipeline (based on a logistic classifier in the tangent space) as in Collas, Ju, et al. (2026) and detail it in Appendix B.3.2.

5.6.2 The results

We report the results in Table 5.4 for the distribution quality and in Table 5.5 for the downstream classification. Riemannian methods substantially outperform EuclideanCFM

Dataset	Model	Train/Synthetic Dens. (↑)	Train/Synthetic Cov. (↑)	Train/Synthetic W_2 (↓)	Test/Synthetic Dens. (↑)	Test/Synthetic Cov. (↑)	Test/Synthetic W_2 (↓)
BNCI2014002	EuclideanCFM	$0.018_{\pm 0.022}$	$0.105_{\pm 0.107}$	$13.894_{\pm 2.299}$	$0.021_{\pm 0.020}$	$0.082_{\pm 0.070}$	$13.804_{\pm 2.584}$
	DiffeoCFM	$0.661_{\pm 0.047}$	$0.969_{\pm 0.011}$	$2.010_{\pm 0.027}$	$\mathbf{0.411}_{\pm 0.041}$	$0.721_{\pm 0.052}$	$2.421_{\pm 0.061}$
	RiemannianCFM	$0.910_{\pm 0.114}$	$0.992_{\pm 0.004}$	$1.916_{\pm 0.022}$	$0.390_{\pm 0.040}$	$0.795_{\pm 0.045}$	$2.346_{\pm 0.057}$
	WrappedCFM	$\mathbf{0.915}_{\pm 0.088}$	$\mathbf{0.994}_{\pm 0.004}$	$\mathbf{1.910}_{\pm 0.026}$	$0.403_{\pm 0.039}$	$\mathbf{0.801}_{\pm 0.045}$	$\mathbf{2.327}_{\pm 0.062}$
	Real-data reference:	Dens. $0.943_{\pm 0.045}$,	Cov. $0.892_{\pm 0.054}$,	W_2 $2.186_{\pm 0.061}$			
BNCI2015001	EuclideanCFM	$0.058_{\pm 0.037}$	$0.257_{\pm 0.125}$	$6.023_{\pm 0.851}$	$0.006_{\pm 0.000}$	$0.019_{\pm 0.010}$	$6.084_{\pm 0.985}$
	DiffeoCFM	$0.270_{\pm 0.021}$	$0.817_{\pm 0.019}$	$1.674_{\pm 0.012}$	$0.028_{\pm 0.004}$	$0.080_{\pm 0.002}$	$2.095_{\pm 0.008}$
	RiemannianCFM	$0.732_{\pm 0.113}$	$0.972_{\pm 0.018}$	$1.600_{\pm 0.037}$	$\mathbf{0.069}_{\pm 0.001}$	$\mathbf{0.161}_{\pm 0.008}$	$\mathbf{2.019}_{\pm 0.007}$
	WrappedCFM	$\mathbf{0.749}_{\pm 0.092}$	$\mathbf{0.975}_{\pm 0.014}$	$\mathbf{1.584}_{\pm 0.036}$	$0.063_{\pm 0.004}$	$0.155_{\pm 0.026}$	$2.038_{\pm 0.017}$
	Real-data reference:	Dens. $0.049_{\pm 0.002}$,	Cov. $0.130_{\pm 0.003}$,	W_2 $2.073_{\pm 0.005}$			
Textile	EuclideanCFM	$0.381_{\pm 0.032}$	$0.928_{\pm 0.018}$	$1.448_{\pm 0.060}$	$0.464_{\pm 0.041}$	$0.957_{\pm 0.016}$	$1.395_{\pm 0.155}$
	DiffeoCFM	$0.757_{\pm 0.030}$	$0.991_{\pm 0.002}$	$0.697_{\pm 0.002}$	$0.834_{\pm 0.035}$	$0.996_{\pm 0.001}$	$0.757_{\pm 0.002}$
	RiemannianCFM	$\mathbf{0.962}_{\pm 0.017}$	$\mathbf{0.997}_{\pm 0.001}$	$\mathbf{0.678}_{\pm 0.001}$	$\mathbf{1.005}_{\pm 0.024}$	$\mathbf{0.999}_{\pm 0.001}$	$\mathbf{0.742}_{\pm 0.003}$
	WrappedCFM	$0.934_{\pm 0.017}$	$\mathbf{0.997}_{\pm 0.001}$	$0.679_{\pm 0.002}$	$0.978_{\pm 0.031}$	$0.998_{\pm 0.002}$	$0.744_{\pm 0.002}$
	Real-data reference:	Dens. $0.974_{\pm 0.025}$,	Cov. $0.999_{\pm 0.000}$,	W_2 $0.723_{\pm 0.001}$			
BreizhCrops	EuclideanCFM	$0.372_{\pm 0.016}$	$0.925_{\pm 0.005}$	$1.270_{\pm 0.078}$	$0.216_{\pm 0.008}$	$0.626_{\pm 0.011}$	$1.235_{\pm 0.101}$
	DiffeoCFM	$0.875_{\pm 0.015}$	$0.995_{\pm 0.001}$	$0.621_{\pm 0.001}$	$0.524_{\pm 0.010}$	$0.853_{\pm 0.011}$	$0.685_{\pm 0.003}$
	RiemannianCFM	$\mathbf{0.963}_{\pm 0.015}$	$0.997_{\pm 0.001}$	$\mathbf{0.614}_{\pm 0.001}$	$\mathbf{0.585}_{\pm 0.009}$	$\mathbf{0.873}_{\pm 0.009}$	$\mathbf{0.681}_{\pm 0.003}$
	WrappedCFM	$0.939_{\pm 0.006}$	$\mathbf{0.998}_{\pm 0.001}$	$\mathbf{0.614}_{\pm 0.001}$	$0.573_{\pm 0.011}$	$0.869_{\pm 0.007}$	$0.682_{\pm 0.002}$
	Real-data reference:	Dens. $0.965_{\pm 0.012}$,	Cov. $0.969_{\pm 0.006}$,	W_2 $0.684_{\pm 0.002}$			

Table 5.4. Distribution quality of the synthetic SPD matrices. Density and coverage assess fidelity and diversity, W_2 is the Wasserstein-2 distance; each is computed between the synthetic matrices and either the train or the test split of the real data. The italic line closing each block is the *reference*: the same three metrics computed between the real train and real test splits, i.e. the value a perfect model would approach — it is a target, not a competitor. Best model per column in bold; our proposed WrappedCFM is highlighted.

across all datasets on both evaluation protocols, demonstrating the importance of manifold-aware methods. This trend is reflected in both the distributional and downstream evaluation metrics. EuclideanCFM produces larger W_2 distances and lower density scores. For example, on BNCI2014002, EuclideanCFM attains a W_2 distance of 13.894 compared to 2.010 for the Riemannian methods, while its density score decreases from 0.661 to 0.018. These deficiencies are also reflected in downstream classification performance, where EuclideanCFM achieves an F1 score of 0.457 on BNCI2014002 compared to 0.718 for the Riemannian approaches. Among the Riemannian approaches, WrappedCFM and RiemannianCFM exhibit similar performance and both outperform DiffeoCFM in terms of sample quality. For example, on BNCI2015001, DiffeoCFM attains a Train/Synthetic density of 0.270 compared to 0.732 and 0.749 for RiemannianCFM and WrappedCFM, respectively, while on Textile the corresponding scores are 0.757 versus 0.962 and 0.934. Overall, these results demonstrate that modeling the SPD geometry is essential for generative modeling, and that the proposed WrappedCFM achieves performance comparable to RiemannianCFM while consistently improving over DiffeoCFM on distribution quality metrics.

5.7 Conclusion

In this chapter, we continued the work started in Chapter 4 around the probabilistic framework for learning tools on $\mathcal{P}_d$, extending this approach to Flow Matching. Indeed, we built a parametric Riemannian CFM model based on the wrapped Gaussian for the conditional probability paths used in CFM. Depending on the parameterization, one could recover state-of-the-art Riemannian CFM methods, namely, the RiemannianCFM from R. T. Q. Chen and

Dataset	Model	TSTR ROC-AUC (↑)	TSTR F1 (↑)	TRTS ROC-AUC (↑)	TRTS F1 (↑)
BNCI2014002	EuclideanCFM	0.712±0.036	0.457±0.297	0.657±0.053	0.682±0.055
	DiffeoCFM	0.808±0.025	0.720±0.021	0.871±0.018	0.788±0.027
	RiemannianCFM	0.810±0.024	0.718±0.029	0.888±0.032	0.807±0.040
	WrappedCFM	**0.818**±0.030	**0.729**±0.033	**0.893**±0.023	**0.818**±0.026
	Real-data reference (**TRTR**): ROC-AUC 0.827±0.032, F1 0.734±0.038				
BNCI2015001	EuclideanCFM	0.702±0.025	0.655±0.013	0.731±0.018	0.691±0.001
	DiffeoCFM	0.719±0.017	**0.677**±0.032	0.829±0.008	0.757±0.006
	RiemannianCFM	**0.722**±0.018	0.673±0.044	**0.860**±0.013	**0.781**±0.006
	WrappedCFM	0.714±0.016	0.651±0.044	0.853±0.013	0.771±0.016
	Real-data reference (**TRTR**): ROC-AUC 0.734±0.014, F1 0.675±0.037				
Textile	EuclideanCFM	0.968±0.003	0.908±0.009	0.945±0.007	0.875±0.016
	DiffeoCFM	0.972±0.003	0.917±0.005	0.966±0.004	0.900±0.013
	RiemannianCFM	**0.973**±0.002	**0.919**±0.004	**0.976**±0.003	**0.919**±0.010
	WrappedCFM	**0.973**±0.002	**0.919**±0.002	0.975±0.003	0.913±0.012
	Real-data reference (**TRTR**): ROC-AUC 0.975±0.002, F1 0.922±0.004				
BreizhCrops	EuclideanCFM	0.962±0.001	0.778±0.002	0.863±0.039	0.605±0.054
	DiffeoCFM	0.974±0.000	0.825±0.003	0.970±0.005	0.803±0.018
	RiemannianCFM	**0.975**±0.000	**0.829**±0.002	**0.984**±0.002	**0.856**±0.010
	WrappedCFM	**0.975**±0.000	0.826±0.001	0.981±0.003	0.845±0.015
	Real-data reference (**TRTR**): ROC-AUC 0.977±0.000, F1 0.853±0.001				

Table 5.5. Downstream classification performance. A logistic regression in the tangent space is trained and tested under three regimes: **TSTR** trains on synthetic data and tests on real data, **TRTS** trains on real data and tests on synthetic data, and **TRTR** trains and tests on real data. **TRTR** involves no synthetic data at all, so it is the upper bound the other two regimes are aiming at; it is reported as the italic reference line closing each block rather than as a competing row. Best model per column in bold; our proposed WrappedCFM is highlighted.

Lipman (2024) and the DiffeoCFM. This extension of the probabilistic framework helps at showing the power of such a framework, as a large class of algorithms can be fitted inside.

Moreover, as we did in Chapter 4 with the *Wrapped Discriminant Analysis*, the *Wrapped Gaussian Kernel* and the *Wrapped Potato* that we started to sketch (and that will be studied deeply in Chapter 6), this framework around the wrapped Gaussian can also help us to build new tools. This is also what we have done in this chapter, by building and studying the *WrappedCFM*. The main ingredient in these new algorithms is that the anchor of the wrapped Gaussian is now a true parameter that can change over time (in the case of the WrappedCFM). Indeed, combined with a non-zero dispersion matrix (unlike the RiemannianCFM), this allows the WrappedCFM to rely on a different tangent space at each time t, therefore a better local approximation of the manifold compared to DiffeoCFM (which considers only one tangent space at the identity).

We also validated our approach through experiments on both synthetic and real-world datasets. Our results demonstrate that geometry-aware, infinitely supported conditional paths are able to sample diverse samples while maintaining high fidelity. While we do not claim a clear superiority compared to RiemannianCFM, we showed that our approach could compete with this state-of-the-art approach. Moreover, it is clear from our experiments that the proposed WrappedCFM is superior to DiffeoCFM, especially when the distribution becomes ill-conditioned. This work continues answering question 2 from the introduction, by building good learning tools relying on the wrapped Gaussian for the manifold $\mathcal{P}_d$.

A lot of extensions of this work are possible, as we have explored the classical and easiest setting of Flow Matching in this chapter. A first question, similar to the one asked at the end of Chapter 3 on wrapped Gaussian, is the extension of WrappedCFM to the broader class of

Hadamard manifolds (Shiga 1984). The wrapped Gaussian can be defined similarly as on $\mathcal{P}_d$ (see Section 3.4), thus, the conditional probability path given in Equation (5.9) can also be defined similarly. The flow given in Equation (5.13) can also be extended in a slightly more complex way (involving the parallel transport in order to move $\bar{\boldsymbol{X}}_t(\boldsymbol{Z})$). However, when it comes to computing the vector field defining the flow as done in Theorem 5.5, one must start by differentiating the flow. In the general case of a Hadamard manifold, this will involve the differentiation of the parallel transport, and, in full generality, the computations get quite convoluted. Using the closed-form formula for the manifold $\mathcal{P}_d$, we were able to carry out those computations, and there may exist other manifolds where this wrapped framework for CFM could lead to interesting results.

Euclidean flow matching and other flow-based models have attracted a lot of attention recently (Michael S. Albergo et al. 2023; X. Liu et al. 2023; Pooladian et al. 2023; A. Y. Tong et al. 2024; Peyré 2026), and there are surely some extensions that could be useful for flow matching on $\mathcal{P}_d$. We could for example think of new parameterizations of our general framework, that could lead to interesting new models. We could also imagine using such Riemannian Flow Matching models to do data augmentation on BCI datasets. Once trained, the FM model is capable of sampling from a given distribution, thus capable of augmenting a given dataset. In order to work, this supposes that there is enough training data available to correctly train the FM model. The work of (Aristimunha, Camargo, et al. 2023) started to investigate how generative models (diffusion models in their case) could help with data scarcity in BCI.

Transition: from theory to applications

Recall the main question we aim at answering in this thesis:

> *Question* 1. What geometry-aware statistical models yield useful representations on the manifold of SPD matrices?

In the first chapters of this thesis, we have started to give an answer to this question. Indeed, in Chapter 3, we introduced the main theoretical tool developed and used in the different works achieved during this PhD: the *wrapped Gaussian*. This distribution is a geometry-aware version of the multivariate Gaussian distribution that is defined on the manifold $\mathcal{P}_d$ of SPD matrices. We showed that this distribution is well-grounded, with interesting theoretical properties, but can also be used in practice, as it has a simple sampling algorithm and a closed-form density.

However, we did not answer an important part of question 1: we want to build a **useful** statistical model on $\mathcal{P}_d$. In Chapter 3, we mostly focus on theoretical properties. We also showed that, given a sufficient amount of points, it was possible to estimate the parameters of the wrapped Gaussian. However, this does not show how this distribution could be useful. In Chapter 4 and Chapter 5, we deeply relied on the wrapped Gaussian to build new tools for learning on the manifold $\mathcal{P}_d$. We showed that the wrapped Gaussian was a good theoretical tool for unifying already existing learning tools such as classifiers, kernels or Flow Matching methods. We also showed that the existing tools could be extended using the different properties of the wrapped Gaussian. The main property used to define new algorithms (the *Wrapped Discriminant Analysis* classifiers or the *Wrapped Gaussian Kernel* from Chapter 4 and the *Wrapped Conditional Flow Matching* from Chapter 5) is that the anchor of the wrapped Gaussian is now a true parameter of the statistical model. Thus, it can be learned, be different between the classes or modified iteratively while still forming a consistent model. This helped us give a first answer to question 2: the wrapped Gaussian can be used to build new tools for learning on $\mathcal{P}_d$, and can also help us unite already known tools in a probabilistic framework.

Nevertheless, at this point, we did not tackle the last question 3: can the wrapped Gaussian tackle variabilities in BCI? In fact, in the numerical experiments we led in Chapter 4 and Chapter 5, we used BCI datasets, and trained classifiers or Flow Matching algorithms on those. But we never studied the real impact of the wrapped Gaussian on those datasets. We did not open the hood of those datasets and looked at the variabilities and how to tackle them. This is the goal of the second part of this manuscript. Now that we have a theoretical tool — the wrapped Gaussian — and some algorithms build upon it, how do they interact with real-life BCI datasets, full of noise and unknown variabilities? Is the flexibility of the wrapped Gaussian capable of learning and better handling variabilities? Can we maybe learn more about those variabilities?

Bridging the gap between the theoretical world and the practical world is a difficult task. In the next chapters, we will give our best shot at it in order to answer question 3.

Part II

Application to Brain-Computer Interfaces

6

Chapter 6

One-Class Classifiers for Anesthetic State Detection

Contents

6.1 Introduction . . . 133
6.2 Anesthetic states and Median Nerve Stimulation . . . 135
6.2.1 Accidental Awareness during General Anesthesia . . . 135
6.2.2 Median Nerve Stimulation . . . 136
6.2.3 Description of the dataset . . . 137
6.3 One-Class Classifiers . . . 137
6.3.1 One-Class MDM . . . 137
6.3.2 One-Class WG . . . 138
6.4 Main Results . . . 139
6.5 Ablation study . . . 140
6.5.1 Influence of the number of electrodes . . . 140
6.5.2 Influence of the type of the dispersion matrix Σ . . . 141
6.5.3 Influence of the threshold ε . . . 142
6.6 Discussion . . . 142
6.6.1 Continuous validation of one-class model scores through different levels of anesthetic depth . . . 142
6.6.2 Robustness in the clinical anesthesia setting . . . 144
6.6.3 Isotropy vs anisotropy . . . 144
6.7 Conclusion . . . 145

6.1 Introduction

In 1987, Box and Draper (1987) wrote "Essentially, all models are wrong, but some are useful", arguing that while no model is perfectly accurate, simple models can still lead to valuable insights. The brain is obviously not a simple system that can easily be modeled. Yet, taking inspiration from Box's saying, we might at least try to fit a relatively simple model of the brain activity, and see if we are nevertheless able to get back some interesting information. In the previous chapters, we have studied a new model for SPD matrices: the wrapped Gaussian in Chapter 3 that we have applied to learning tools on the manifold $\mathcal{P}_d$ of SPD matrices in Chapter 4 and Chapter 5. In this thesis, the motivation of studying SPD matrices is

that the main feature we use when building BCIs is the covariance matrix estimated from an EEG. The goal of this chapter is therefore to see if the theory we have developed in the previous chapters can be useful when dealing with real-life EEGs. For this, we had the chance to collaborate with another PhD student, Valérie Marissens Cueva, who also works on Riemannian techniques for BCI and who has collected real-life EEG data. The goal of this data collection was to record patients' brain activity during general anesthesia using EEG in order to detect possible *Accidental Awareness during General Anesthesia* (AAGA). AAGA is an unintended episode of consciousness during surgery under general anesthesia that may result in significant psychological distress for patients. Thus, detecting them as soon as they happen is crucial, and recent promising methods to do so are based on BCIs (Rimbert, Riff, et al. 2019; Blokland et al. 2016; Marissens Cueva, Rimbert, Alvarez, et al. 2025). Many anesthetic drug regimens include neuromuscular blocking agents, which prevent the patient from moving and therefore from alerting clinicians if they regain consciousness. This is one of the motivations for using EEG and BCIs: they can detect motor-related brain activity or neural markers of consciousness even in the absence of any behavioral signs.

The opportunity of working on such a dataset was unique, as we could have access to data that were **not** recorded in a controlled laboratory, with a very controlled setup, as is the case for the datasets we were used to, such as the ones used in Chapter 4 or Chapter 5. Here, the data was acquired in a real surgical environment, in a hospital which inevitably leads to multiple sources of variabilities, including multiple electromagnetic noises in the operating room, variations in patient positioning impacting EEG signal stability, heterogeneous anesthetic regimens, and inter-subject variability. Thus, the goal of this chapter is to study if our "simple" model, the wrapped Gaussian, could be useful in differentiating between "awake" and "anesthetized", which is a first step in better detecting AAGA. A classical approach (Marissens Cueva, Rimbert, Alvarez, et al. 2025) is based on the *Minimum Distance to the Mean* algorithm (Barachant, Bonnet, et al. 2010b), and as we have seen in Chapter 4, this approach is based on an isotropic assumption. We will see that, by leveraging the anisotropy of the wrapped Gaussian, we are able to better distinguish between a patient being awake and a patient being anesthetized. This is a promising step towards detecting AAGA.

One-Class Classification

Another difficulty of detecting AAGA is that before the surgery starts, we only have access to EEGs recorded when the patient was awake. Thus, we only have access to one of the two classes of our two-class classification problem (`awake` vs `anesthetized`). This calls for the use of *One-Class Classifiers* (Perera et al. 2021). In our case, we will only be able to train on trials that all belong to the class `awake`. Once the surgery has started, the goal is that our classifier correctly classifies the trials as `anesthetized`, except when the patient is experiencing AAGA: we want to detect rare deviations from a baseline state. In this dataset, we do not have real AAGA happening, thus, the evaluation will consist in differentiating between `awake` and `anesthetized`. If we are able to correctly do this, a similar BCI could be used in the case of a real AAGA, as it constitutes a first promising step. For this study, everything has been done *a posteriori*, the training and the testing have been performed after the dataset was recorded.

Other problems in BCI also require the use of One-Class Classifiers. For example, BCIs have been used to non-invasively detect anomalous respiratory states (Navarro-Sune et al. 2016). For this application, a One-Class Classifier is trained on the trials during comfortable breathing and the goal is to detect motor cortex activation provoked during uncomfortable breathing. One-Class Classifiers can also be used to detect epileptic seizures (Tran et al. 2022) or sleep apnea (L. D. Barnes et al. 2022).

More generally, the goal of a One-Class Classifier is to learn a useful representation of the data that enables recognition of positively labeled queries during inference (Perera et al. 2021).

One-Class classification methods have been used to detect abnormal events, such as outlier images or events, and are thus related to outlier detection. We have already introduced a Riemannian outlier detection algorithm in Section 4.4, the *Riemannian potato* (Barachant, Andreev, et al. 2013; Barthélemy et al. 2019). Moreover, in Remark 4.11, we started to introduce a wrapped version of the Riemannian potato, taking advantage of the anisotropy of the wrapped Gaussian. The Riemannian potato was developed for online outlier detection, so we will see how one can transform it into a One-Class Classifier, that we call the *One-Class MDM* (we will see that, unsurprisingly, it is closely related to the MDM). We will then propose a new One-Class Classifier, that we call the *One-Class WG*, that leverages the wrapped Gaussian from Chapter 3 to build a more robust classifier.

Outline of the chapter

We will start by introducing Accidental Awareness during General Anesthesia, as well as a promising BCI approach that could be used to potentially detect them based on the stimulation of the median nerve in Section 6.2. We will also introduce the dataset used for this study at the end of this section. Then, we will introduce the two One-Class Classifiers that we will use in Section 6.3. We will give the main results of the different classifiers considered in Section 6.4 as well as an ablation study in Section 6.5, where we have studied the effect of several hyperparameters of the One-Class WG. Finally, we discuss the results in Section 6.6, show how they correlate to the concentration of anesthetic (here propofol) provided to the patient and explain quantitatively why the One-Class WG outperforms the One-Class MDM.

This work was done in collaboration with another PhD student, Valérie Marissens Cueva and led to a publication at the Graz BCI Conference:

> Valérie Marissens Cueva*, **Thibault de Surrel***, Laurent Bougrain, Seyed Javad Bidgoli, Guy M Cheron, Ana Maria Cebolla Alvarez, Claude Meistelman, Fabien Lotte, Florian Yger, and Sébastien Rimbert (2026). "Wrapped One-Class Riemannian EEG Classifier for BCI-Based Detection of Anesthetic States". In: *10th International Graz Brain-Computer Interface Conference*

∗ These two authors contributed equally to this work.

6.2 Anesthetic states and Median Nerve Stimulation

For this study, we focus on a specific dataset collected by my co-author Valérie Marissens Cueva. The goal of this dataset is to record the brain's activity during general anesthesia using EEG in order to explore how to possibly detect *Accidental Awareness during General Anesthesia* (AAGA). In this section, we will start by introducing AAGA, as well as the consequences it can have on patients. Then, we will explore a promising approach based on BCI to potentially detect them. Finally, we will present in more details the dataset we will use in this chapter.

6.2.1 Accidental Awareness during General Anesthesia

Accidental Awareness during General Anesthesia (AAGA) is an unexpected awakening of the patient during a surgical procedure under general anesthesia (J.J. Pandit et al. 2014; Almeida 2015). This situation occurs when the depth of anesthesia is not enough to compensate for surgical and environmental stimuli. The patient may then experience pain and later recall events related to the surgery. This awakening can lead to potentially severe psychological sequelae, such as Post-Traumatic Stress Disorder (PTSD), anxiety, insomnia, flashbacks, lack of confidence in the medical staff, among other effects (Leslie et al. 2010; Schwender et al. 1995;

JJ Pandit and Cook 2013; Almeida 2015). Moreover, AAGA also generates a high anxiety level in anesthesiologists (L. Xu et al. 2009). The real number of patients experiencing AAGA is debatable, as necessary questionnaires following the surgery are not always available, however, studies estimate that AAGA occurs in about 1% of high-risk interventions (Tasbihgou et al. 2018; JJ Pandit and Cook 2013). Finding a way to detect and prevent AAGA is thus critical. A first naive idea could be to detect the patient's movement, as the first reaction of a patient feeling some pain when anesthesia depth is insufficient would be to move. However, anesthesia is typically paired with the use of Neuromuscular-Blocking Agents that paralyze the patient (Bruhn et al. 2006). Moreover, traditional clinical features, such as heart rate, blood pressure or sweating are also unreliable AAGA detectors (Bruhn et al. 2006; Punjasawadwong et al. 2014).

6.2.2 Median Nerve Stimulation

The use of EEG to monitor the depth of anesthesia has led to encouraging results (Purdon et al. 2015). Indeed, trying to detect the first reflex of moving from a patient experiencing AAGA could be done using a BCI based on motor imagery (Rimbert, Riff, et al. 2019; Blokland et al. 2016; Marissens Cueva, Rimbert, Alvarez, et al. 2025). However, most motor imagery-based BCIs rely on subjects imagining movements following **explicit** visual or auditory cues, a paradigm that differs substantially from the conditions encountered during surgery. Thus, strategies must be used to design a passive BCI that is able to detect the **implicit** intention of movement of a patient experiencing AAGA. This comes with another challenge: detecting intention of movements without any time markers. This means continuously analyzing the EEG signal without any information on when to look for a specific pattern, which is known as asynchronous BCIs. In traditional motor imagery-based BCIs, the scientist knows when the cue was given, which is not the case here. In order to propose a solution satisfying those requirements, Rimbert, Riff, et al. (2019) propose to use the *Median Nerve Stimulation* (MNS). This consists in painlessly stimulating the median nerve, the main nerve of the front of the forearm, also called the "eye of the hand", that controls the thumb abduction, wrist flexion, and flexion of the finger phalanges (Stretanski et al. 2026). In parallel, the brain activity of the patient is recorded using EEG. The setup is illustrated in Figure 6.1. It has been shown that discriminating between MNS alone and MNS **combined with motor intention** is more effective than distinguishing motor intention from rest (Rimbert, Riff, et al. 2019; Avilov et al. 2021). Also, MNS intrinsically provides a trigger to know when to analyze the signal, leading to an increase in performances compared to asynchronous BCIs (Rimbert, Lelarge, et al. 2023).

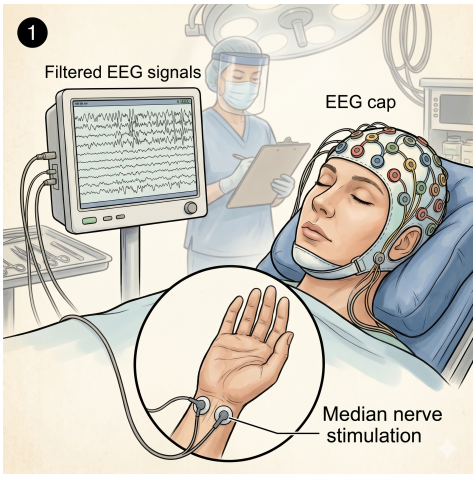


Figure 6.1. Illustration of the Median Nerve Stimulation (MNS).
Generated using Nano Banana 2.

Our objective is thus to differentiate EEG responses recorded in awake subjects from those observed under altered brain dynamics induced by general anesthesia. Ultimately, this framework aims to contribute to the detection of intraoperative awareness by identifying potential motor intention from the anesthetized patient. This setting naturally defines a one-class learning problem, as only the reference condition can be collected before the surgery (here EEG during the awake state) for patient-specific calibration, while the alternative condition (here patient-specific EEG during anesthesia) is not accessible beforehand. Beyond this specific use case, the proposed method could be relevant for a broader range of BCI

applications involving reference-only training, such as passive BCIs, vigilance monitoring, or brain-state change detection (Ruff et al. 2021).

6.2.3 Description of the dataset

In the dataset collected by Valérie Marissens Cueva at the hospital of Brugmann in Belgium, a total of 28 patients scheduled for surgery under general anesthesia were initially enrolled. Four patients were excluded either due to technical or logistical reasons beyond our control. Five additional subjects were excluded to ensure dataset homogeneity in terms of stimulation side and anesthesia protocol. The final cohort comprised 19 patients, including 10 females, with an average age of 47.47 ± 12.51 years.

The anesthetic used was propofol. During the surgeries, the MNS consisted of 0.2 ms square electrical pulses, delivered to the left wrist with a 3-4 s inter-stimulus interval. Stimulation intensity was individually adjusted to elicit a visible thumb twitch (Schnitzler et al. 1997) and remained below 15 mA. The EEGs were then recorded using a 64-channel cap. The data was downsampled to 128 Hz and band-pass filtered (8-30 Hz). Artifacts were then visually rejected, and the data was epoched into 750ms windows (250-1000 ms after MNS). Each run had 150 MNS across two sessions, one recorded before the surgery (preoperative session) and one recorded during the surgery (intraoperative session). In the preoperative session, at least one run was acquired with the awake patient, yielding 94-499 MNS trials per subject. During the intraoperative session, several EEG runs were recorded, the number of runs depending on the surgery duration, resulting in 415-2390 MNS trials in the anesthetized condition. Moreover, the target propofol concentrations were recorded simultaneously with the EEG signals. Finally, from each trial we estimated the spatial covariance matrix, regularizing them using the shrinkage estimator from Ledoit and Wolf (2004) in order to avoid any ill-conditioned covariance matrix.

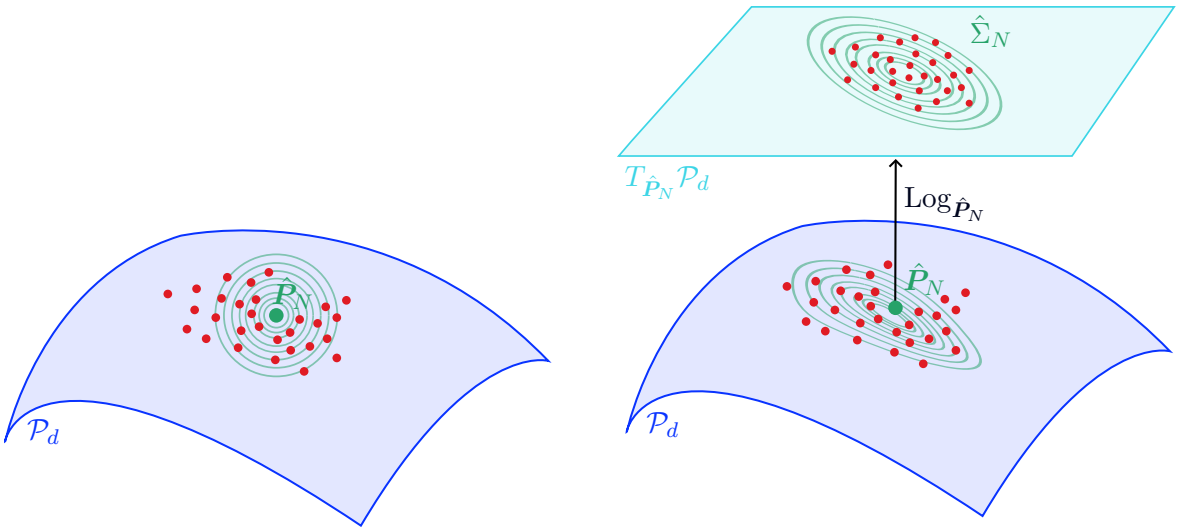
6.3 One-Class Classifiers

Let us now introduce the two One-Class Classifiers that we will compare in this study: the *One-Class MDM* already studied in Marissens Cueva, Rimbert, Alvarez, et al. (2025) and which can be seen as an offline version of the Riemannian potato that was studied at Section 4.4, and the proposed *One-Class WG*, an anisotropic One-Class classifier that relies on the wrapped Gaussian. The two One-Class classifiers described here are illustrated in Figure 6.2.

6.3.1 One-Class MDM

Before introducing the first One-Class Classifier that we consider, let us first see how the *Riemannian potato* from Barachant, Andreev, et al. (2013) is related to the MDM classifier (Barachant, Bonnet, et al. 2011) studied in Section 4.3.1. As we saw in Chapter 4 (which is summarized in Figure 4.5), both algorithms rely on the isotropic Gaussian (Said, Bombrun, et al. 2017) and both take their decision by computing the likelihood under a certain isotropic Gaussian. The main difference is that the Riemannian potato is an online tool, that updates its prototype (the Fréchet mean), when a new sample arrives.

The first One-Class Classifier we consider draws inspiration from both those algorithms, and is called the *One-Class MDM*. It was studied to measure anesthetic states in Marissens Cueva, Rimbert, Cebolla Alvarez, et al. (2024) and Marissens Cueva, Rimbert, Alvarez, et al. (2025). Given a training dataset $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ of N SPD matrices of the class `awake`, the One-Class MDM computes their Fréchet mean $\hat{\boldsymbol{X}}_N$. Then, given a new SPD matrix $\boldsymbol{Z}$, it computes the AIRM distance $\delta(\boldsymbol{Z}, \hat{\boldsymbol{X}}_N)$ between $\boldsymbol{Z}$ and $\hat{\boldsymbol{X}}_N$ and classifies $\boldsymbol{Z}$ as `awake` if $\delta(\boldsymbol{Z}, \hat{\boldsymbol{X}}_N) \leqslant \varepsilon$ and `anesthetized` otherwise, where ε is a threshold. In the experiments, the



One-Class MDM One-Class WG

Figure 6.2. Illustration of the two compared algorithms: *One-Class MDM* (left) and *One-Class WG* (right). While *One-Class MDM* estimates only the Fréchet mean $\hat{\boldsymbol{P}}_N$ of the training covariance matrices $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, $\boldsymbol{X}_3$,..., *One-Class WG* also estimates a dispersion matrix in the tangent space $T_{\hat{\boldsymbol{P}}_N}\mathcal{P}_d$.

threshold ε was set to the same threshold as in the Riemannian potato described in Section 4.4, that is:

$$\varepsilon = \mu + 2.5\sigma \text{ where } \mu = \frac{1}{N}\sum_{i=1}^{N}\delta(\boldsymbol{X}_i, \hat{\boldsymbol{X}}_N) \text{ and } \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\delta(\boldsymbol{X}_i, \hat{\boldsymbol{X}}_N) - \mu)^2}.$$

Remark 6.1. Similarly to the Riemannian potato in Proposition 4.10, the decision rule can be rewritten as a likelihood threshold under the isotropic Gaussian $G(\hat{\boldsymbol{X}}_N, \sigma)$: $\boldsymbol{Z}$ is classified as `awake` if $\mathcal{L}_{\hat{\boldsymbol{X}}_N,\sigma}(\boldsymbol{Z}) \geqslant \ell_\varepsilon$ and `anesthetized` otherwise, where $\mathcal{L}_{\hat{\boldsymbol{X}}_N,\sigma}$ is the likelihood of $G(\hat{\boldsymbol{X}}_N, \sigma)$ and ℓ_ε is the same threshold as in Proposition 4.10.

Remark 6.2. As we said for the Riemannian potato in Chapter 6, this One-Class classifier relies on an *isotropic assumption*, the spread of the data is thus assumed to be uniform across all spatial directions. Indeed, the classification boundary of One-Class MDM is the sphere centered at $\hat{\boldsymbol{X}}_N$ and of radius ε:

$$\mathcal{R}_{\texttt{awake}} = \{\boldsymbol{Z} \in \mathcal{P}_d \mid \delta(\boldsymbol{Z}, \hat{\boldsymbol{X}}_N) \leqslant \varepsilon\}.$$

Moreover, this method is an *intrinsic method*, using the categories described in Section 2.3.6.

6.3.2 One-Class WG

Let us now introduce a novel one-class classifier, called the *One-Class WG*, based on the Wrapped Gaussian distribution introduced in Chapter 3. This classifier does not rely solely on the Riemannian mean of the training data and does not assume any isotropy of the data. Given N SPD matrices $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ from the `awake` class, our classifier fits a wrapped Gaussian $\text{WG}(\boldsymbol{P}; 0_D, \Sigma)$ anchored at $\boldsymbol{P}$, with a zero mean in the tangent space and with a dispersion matrix Σ in the tangent space. Let us make some remarks on the choice of this distribution.

Remark 6.3. In Chapter 3, we have argued that the wrapped Gaussian distribution we present is strictly more expressive than variants that impose $\mu = 0$ (the mean in the tangent space), as in Galaz-Garcia et al. (2022) and Nagano et al. (2019) (see Proposition 3.25).

While this is true from a theoretical point of view, things get difficult when wanting to estimate this full model from real data. Indeed, we saw in Section 3.3, and especially in the numerical experiments, that estimating a full wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ from a finite sample required a lot of data (see Figure 3.5). Here, we do not have enough data to correctly fit a full model. This is why we decided to restrict ourselves to having a centered distribution in the tangent space (i.e. $\mu = 0$). Moreover, this choice solves the identifiability issue we had in Section 3.2.4. The wrapped Gaussian model $\{\mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma), \boldsymbol{P} \in \mathcal{P}_d \text{ and } \Sigma \in \mathcal{P}_D\}$ is identifiable.

In practice, and using Section 3.3.2, the One-Class WG estimates the parameters $\boldsymbol{P}$ and Σ of the wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$ using the Fréchet mean $\hat{\boldsymbol{X}}_N$ for $\boldsymbol{P}$ and the Euclidean MLE for the dispersion matrix Σ in the tangent space at $\hat{\boldsymbol{X}}_N$ (see Proposition 3.35). For a new SPD matrix $\boldsymbol{Z}$, the log-likelihood of $\boldsymbol{Z}$ under $\mathrm{WG}(\hat{\boldsymbol{X}}_N; 0_D, \hat{\Sigma}_N)$ is computed, and the trial is classified as `awake` if it exceeds a given threshold ε and `anesthetized` otherwise. In other words, by denoting $\hat{y}_{\boldsymbol{Z}}$ the decision taken for trial $\boldsymbol{Z}$, one has

$$\hat{y}_{\boldsymbol{Z}} = \begin{cases} \texttt{awake} & \text{if } \ell(\boldsymbol{Z}) \geqslant \varepsilon, \\ \texttt{anesthetized} & \text{if } \ell(\boldsymbol{Z}) < \varepsilon, \end{cases}$$

where $\ell(\boldsymbol{Z}) = \ell_{\hat{\boldsymbol{X}}_N, 0_D, \hat{\Sigma}_N}(\boldsymbol{Z})$ is the log-likelihood of sample $\boldsymbol{Z}$ under the $\mathrm{WG}(\hat{\boldsymbol{X}}_N; 0_D, \hat{\Sigma}_N)$ distribution. In the experiments, the threshold ε was set to the 95-th percentile of the training log-likelihoods (this value will be challenged in the following experiments). A low ε makes the classifier more permissive, reducing false alarms on `awake` epochs at the cost of missing `anesthetized` ones; a high ε increases sensitivity to anesthesia but incorrectly rejects more `awake` samples.

Remark 6.4. Unlike the One-Class MDM, this new One-Class classifier, the One-Class WG, does not assume isotropy of the data. Indeed, by estimating the dispersion matrix Σ in the tangent space at $\boldsymbol{P}$, the classifier produces a decision boundary that deviates from a perfect sphere, being stretched or compressed along directions encoded by Σ and the Jacobian determinant of the exponential map. Moreover, this method is an *extrinsic method* as it relies on the tangent space at the Fréchet mean.

6.4 Main Results

Setup The two One-Class Classifiers were trained on the first 65 preoperative trials, the `awake` class, and evaluated on the remaining preoperative trials (`awake` test class) and on the entire intraoperative set (`anesthetized` test class), resulting in 444-2472 test trials per subject. Thus, in the training dataset, only the `awake` class appears, while in the testing dataset, both classes `awake` and `anesthetized` are present.

Results Let us start by comparing the mean balanced accuracy across all subjects of both classifiers One-Class MDM and One-Class WG on the data described in Section 6.2.3. The choice of this metric was driven by the highly asymmetric class imbalance. One-Class MDM achieved $78.67\% \pm 15.93$, whereas One-Class WG reached $83.88\% \pm 11.69$ (see Figure 6.3). A two-sided

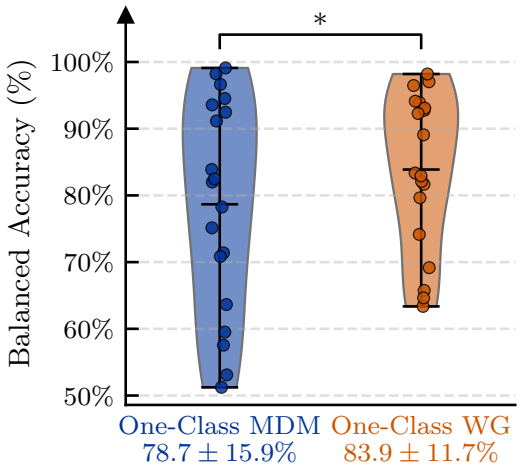


Figure 6.3. Distribution of the balanced accuracy across all subjects using 5 electrodes. A two-sided paired t-test shows that One-Class WG significantly outperforms One-Class MDM ($^*p < 0.05$). Each circle is a subject.

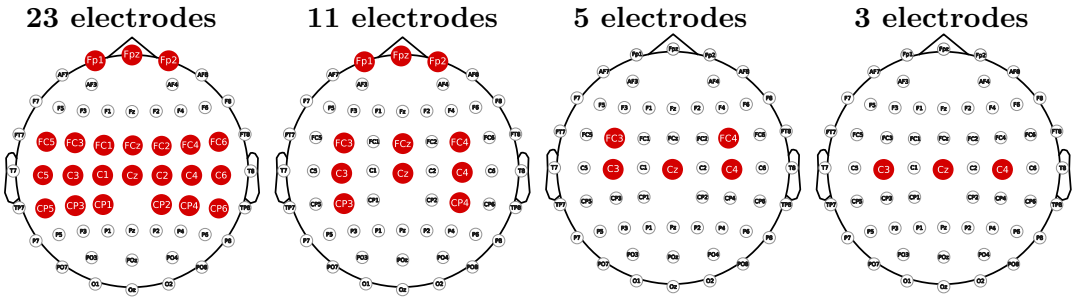


Figure 6.4. Electrode layouts tested compared to the full 64 electrodes montage. The electrodes in red are the ones selected for each setup.

paired t-test confirmed that One-Class WG significantly outperformed One-Class MDM ($p = 0.0118$). Looking subject-wise, for 13 subjects out of the 19 subjects, One-Class WG substantially improves the performances. These results were obtained when considering only five electrodes (C3, C4, Cz, FC4 and FC3) and for the One-Class WG, the estimated dispersion matrix $\hat{\Sigma}_N$ is only diagonal. We will conduct an ablation study on the number of electrodes, on whether $\hat{\Sigma}_N$ should be diagonal, and on the choice of the log-likelihood threshold in Section 6.5.

Nevertheless, we can say that in this specific setup, the proposed One-Class WG performs better than the classical One-Class MDM. This is not very surprising as the One-Class WG takes into account any potential anisotropy in the data. Thus, the One-Class WG is a more expressive model than the isotropic One-Class MDM.

6.5 Ablation study

For this experiment, we have several parameters that we can tweak: the number of electrodes we consider, whether the dispersion matrix Σ of the wrapped Gaussian used to model the data in the One-Class WG is diagonal or not, and the choice of the threshold ε on the log-likelihood for the One-Class WG. In the previous section, we have given the best results we were able to achieve after carefully choosing the hyperparameters. In this section, we present the ablation study of the different parameters.

6.5.1 Influence of the number of electrodes

In this dataset, the EEGs were recorded using 64 electrodes; this means that the spatial covariance matrices are 64×64 SPD matrices. With the One-Class WG the estimated dispersion matrix $\hat{\Sigma}_N$ will be of dimension $2,080 \times 2,080$ (as $64 \times (64+1)/2 = 2,080$). The small number of training data (65 trials) and their short duration of 750 ms are not sufficient to correctly estimate $\hat{\Sigma}_N$. This is why we tried different montages with a lower number of electrodes. More specifically, we studied five different setups with a decreasing number of electrodes: the full setup with 64 electrodes, then a setup with only 23, one with 11 and two very restricted ones containing only 5 and 3 electrodes. The different setups are illustrated in Figure 6.4. Current anesthesia monitors use frontal electrodes to detect changes in consciousness associated with general anesthesia (Purdon et al. 2015); this is why for the setups with 23 and 11 electrodes, we kept the frontal electrodes Fp1, Fpz, Fp2. However, the stimulation of the median nerve evokes a motor task, thus the electrodes located in the motor cortex are also of interest. We thus kept the main electrodes located over the sensorimotor cortex (C3, Cz and C4) for all the setups, extending the sensorimotor coverage as we considered more electrodes.

The results for the different setups, as well as the significance level of the two-sided paired

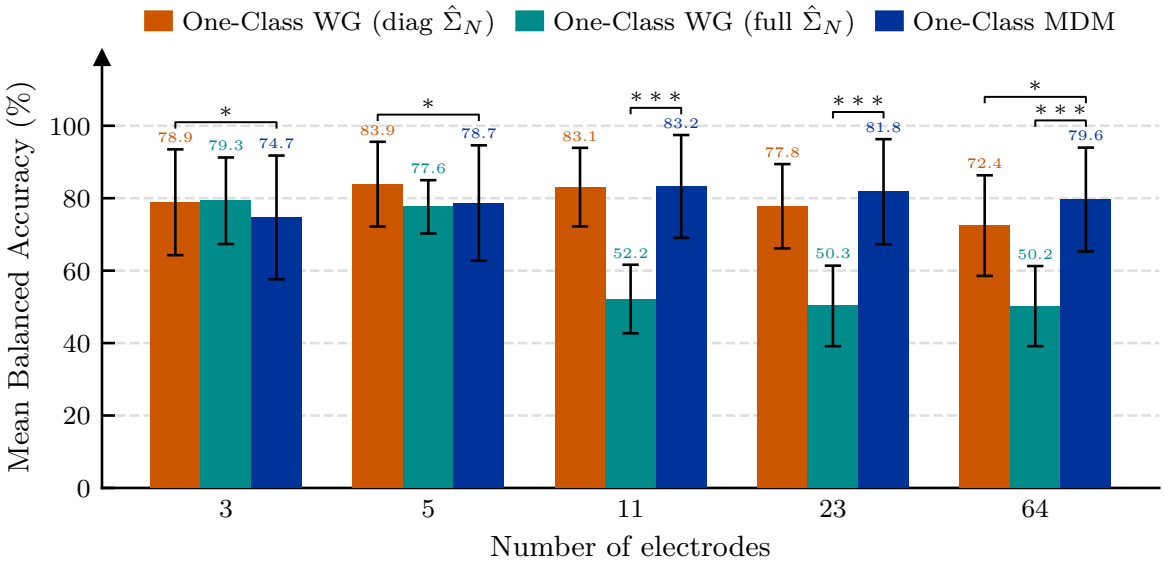


Figure 6.5. Mean test balanced accuracy of the different One-Class classifiers across different electrode setups. Statistical significance was assessed using a two-sided paired t-test (*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$).

t-test between One-Class WG and One-Class MDM are given in Figure 6.5. One can see that, when the number of electrodes considered rises, the performance of the One-Class WG significantly drops, even with a diagonal dispersion matrix $\hat{\Sigma}_N$ (which we get into below). This is as expected. For the One-Class MDM, the performances start by increasing when increasing the number of electrodes, up to 83.2% with 11 electrodes, and then starts falling down again. For the proposed One-Class WG, the best number of electrodes is 5, while for the One-Class MDM, the best number of electrodes is 11. With more electrodes, One-Class MDM outperforms our proposed method. However, the overall best score is obtained by the One-Class WG with 5 electrodes (83.9%).

However, this result is interesting for our study of detecting awareness during surgery, as in a real-life surgical setting, simpler setups are preferred. Indeed, fewer electrodes mean less preparation work to install them, thus gaining time in the preoperative phase. Moreover, as shown in our results, the proposed method works best with 3–5 electrodes, improving mean accuracy across all subjects by up to 5% over One-Class MDM. Thus, our proposed One-Class WG is a strong candidate for practical use in surgery, requiring fewer electrodes than its MDM counterpart.

6.5.2 Influence of the type of the dispersion matrix Σ

Another way to mitigate the very low number of training samples we have access to is to put a prior on the shape of the dispersion matrix Σ used in the wrapped Gaussian modeling the data in the One-Class Classifier. We saw in Chapter 3 that considering a diagonal Σ simplifies the estimation problem and reduces the estimation error (see for examples Figure 3.6). Indeed, considering a diagonal Σ reduces the number of parameters to estimate from $d(d+1)/2 \times d(d+1)/2$ to simply $d(d+1)/2$ (where d is the number of electrodes considered). This choice also implies a modeling choice: we do not consider anymore the relationships between the coefficients of the covariance matrices. These can be seen as fourth-order moments with respect to the EEG recording.

This leads to two variants of the proposed One-Class WG: with a full dispersion matrix $\hat{\Sigma}_N$ and with a diagonal one. The results of these two classifiers using the different montages are also given in Figure 6.5. One can see that, as soon as the number of electrodes is at least

5, the performances of the One-Class WG with a diagonal $\hat{\Sigma}_N$ are much better than the one with a full $\hat{\Sigma}_N$. However, when the number of electrodes is low (e.g. 3 electrodes), we have enough training data to correctly estimate the full matrix $\hat{\Sigma}_N$ and thus, the One-Class WG with a full Σ is the best setup.

NEW

6.5.3 Influence of the threshold ε

The last hyperparameter of the One-Class WG is the threshold ε that dictates the shape of the decision boundary. When training the One-Class WG, we compute the log-likelihoods of the training samples and the decision threshold ε is chosen to be certain percentile of those training log-likelihoods. The smaller the threshold, the more covariance matrices will be classified as `awake` as the smaller the "rejection region" will be. For this last ablation study, we considered four different percentiles of the training log-likelihoods: $0.85, 0.90, 0.95$ and 0.99. We set the number of electrodes to 5 and the dispersion matrix Σ to be diagonal. The balanced accuracies for the different thresholds are given in Figure 6.6.

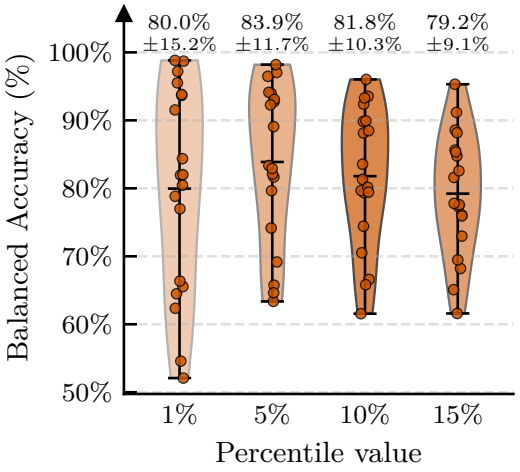


Figure 6.6. Balanced accuracy of the One-Class WG across subjects for four percentile thresholds $\varepsilon \in \{0.85, 0.90, 0.95, 0.99\}$. Setup: 5 electrodes, diagonal dispersion matrix Σ.

One can see that the value of the threshold does not have a huge impact on the classification results (at least for the range of ε chosen). However, the best choice for ε is to take the 0.95-th percentile of the training log-likelihoods.

6.6 Discussion

Overall, the proposed One-Class WG improved discrimination between `awake` and `anesthetized` conditions compared to the One-Class MDM, with reduced electrode montages, despite the variability of a real clinical setting. The following discussion examines these results in terms of continuous behavior of model scores across anesthetic depth, robustness to clinical variability, and the role of anisotropic dispersion modeling.

6.6.1 Continuous validation of one-class model scores through different levels of anesthetic depth

To further understand model behaviors, we analyze the temporal evolution of their respective decision scores in relation to propofol concentration. Rather than focusing only on binary classification accuracy, this analysis allows us to evaluate how each model responds continuously to changes in anesthetic depth. We illustrate the temporal evolution of the model scores for subjects 4 and 24 at Figure 6.7. These are two interesting subjects as the two classifiers work very differently for both of them: for subject 4, the One-Class MDM performs better, with a 99.1% balanced accuracy while the One-Class WG only gives a balanced accuracy of 93.9%. For subject 24, it is the opposite: the One-Class WG achieves a 74.2% score when the One-Class MDM has an accuracy close to chance level (51.2%).

We recall that, for the One-Class WG, the score corresponds to the log-likelihood of each median nerve stimulation-evoked covariance matrix under the fitted Wrapped Gaussian distribution. For the One-Class MDM, the score is the AIRM distance to the `awake` centroid. In Figure 6.7, each circle represents a covariance matrix plotted at its recording time (x-axis)

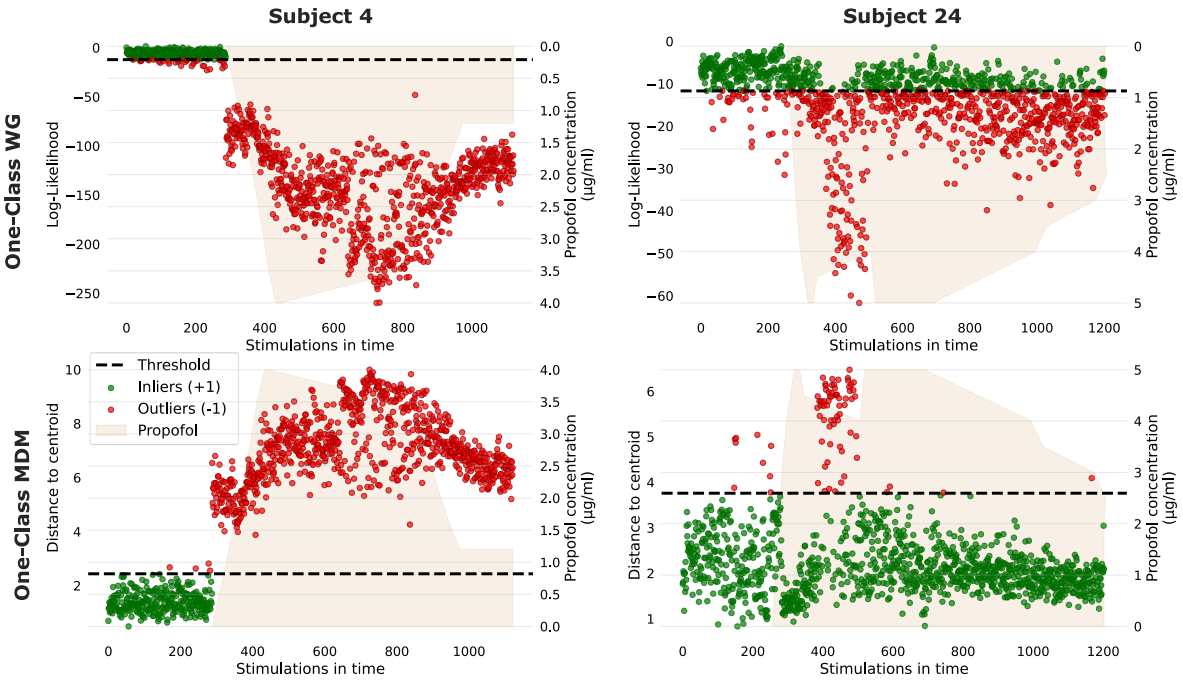


Figure 6.7. Temporal evolution of one-class model scores for two representative subjects. Each circle represents a covariance matrix plotted at its recording time against the classifier score (log-likelihood for One-Class WG; Riemannian distance to the `awake` centroid for One-Class MDM). Propofol concentration is displayed in the background. The decision threshold is indicated by a horizontal line, and colors denote the predicted class (green: `awake`, red: `anesthetized`). Subject 4 (One-Class WG: 93.9%, One-Class MDM: 99.1%). Subject 24 (One-Class WG: 74.2%, One-Class MDM: 51.2%).

against the classifier score (y-axis). Propofol concentration is displayed in the background to assess consistency with anesthetic depth. The plots include both the training set (first 65 `awake` trials), and the testing set (remaining `awake` trials followed by `anesthetized` trials acquired under different concentrations). Finally, the threshold for each One-Class classifier is displayed and the colors correspond to the decision taken by each One-Class classifier (green is `awake` and red is `anesthetized`).

For subject 4 (accuracy One-Class WG $<$ accuracy One-Class MDM), both models clearly separate `awake` and `anesthetized` trials, with scores progressively diverging from the `awake` reference as propofol increases. The slightly lower *One-Class WG* accuracy may reflect a conservative decision threshold, rather than a failure of the underlying distributional modeling. The visible gap between conditions suggests that a Riemannian change-point detection algorithm (X. Wang, R. A. Borsoi, et al. 2024) could be well suited to this setting.

In contrast, subject 24 exhibits the largest discrepancy between the accuracy of both One-Class Classifiers (One-Class WG $>$ One-Class MDM). In this case, both models show increasing deviation from the awake reference as propofol concentration rises beyond 4-5 μg/mL. However, One-Class WG produces a more consistent separation between awake and anesthetized trials, whereas One-Class MDM does not. One possible explanation lies in the EEG dynamics at high propofol concentrations, often associated with burst suppression patterns, which are composed of periods of isoelectric activity and high-amplitude bursts (Pawar and Barreto Chang 2022). Such dynamics induce strong variability in covariance structures. Because One-Class MDM relies solely on distance to a centroid, it does not explicitly model dispersion, whereas One-Class WG models not only the centroid but also the variability of the `awake` distribution through the dispersion matrix, yet covariance matrices corresponding to burst activity may not be uniformly distant from the `awake` centroid. Future analyses could quantify burst suppression via burst suppression ratio (Purdon et al. 2015) and

relate it to one-class model scores, or use Riemannian visualization techniques (such as the ones developed in Chapter 7, see specifically Figure 7.11 for the visualization of the covariance matrices estimated from the EEGs of subject 24) to enable a geometric interpretation of how deep anesthesia shifts the data distribution relative to the `awake` reference. Further work will improve robustness in subjects exhibiting lower classification performance.

6.6.2 Robustness in the clinical anesthesia setting

An important aspect of the proposed approach is its robustness to variability. Unlike controlled laboratory BCI datasets, the data studied here were acquired in a real surgical environment, introducing multiple sources of variability, including operating room noise, patient positioning affecting EEG stability, heterogeneous anesthesia protocols, and inter-subject differences. In addition, preoperative and intraoperative conditions differed slightly, with recordings acquired in different rooms and potentially under different body and arm positions (e.g., supine position and arm placement during surgery versus preoperative setup), which may influence motor-related EEG responses to median nerve stimulation. Such factors could partly facilitate the detection of the `awake` class. Despite this variability, One-Class WG outperformed One-Class MDM, suggesting that explicitly modeling dispersion of the reference class may improve robustness compared to centroid-based methods.

6.6.3 Isotropy vs anisotropy

Let us now make a hypothesis to explain why the proposed One-Class WG performs better than One-Class MDM. The One-Class MDM only estimates the class mean and makes decisions under an isotropic assumption, whereas One-Class WG also estimates a dispersion matrix $\hat{\Sigma}_N$ that captures the anisotropic spread of the point cloud. This anisotropic behavior of the point cloud cannot be captured by the One-Class MDM. Our hypothesis is that it is this anisotropic structure that helps the One-Class WG perform better.

In order to verify this hypothesis, we want to compare the subjects where the One-Class WG leads to better results than the One-Class MDM to the subjects where it is the inverse. More precisely, we want to see whether subjects in the first group (One-Class WG > One-Class MDM) are "more anisotropic" than those in the second group (One-Class MDM > One-Class WG). In order to quantify this, we looked at the variance of the diagonal elements of $\hat{\Sigma}_N$ in the 5-electrodes setup and when $\hat{\Sigma}_N$ is only diagonal (the setup where One-Class WG performs the best). Large variance across diagonal coefficients suggest an anisotropic point cloud, whereas similar diagonal coefficients indicate isotropy. We write the estimated dispersion matrix $\hat{\Sigma}_N$ as follows:

$$\hat{\Sigma}_N = \begin{pmatrix} \hat{\sigma}_1^2 & & \\ & \ddots & \\ & & \hat{\sigma}_{d(d+1)/2}^2 \end{pmatrix}.$$

Then, we compute the variance of the diagonal coefficients $\hat{\sigma}_1^2, ..., \hat{\sigma}_{d(d+1)/2}^2$, yielding one variance per subject. For the first group (One-Class WG > One-Class MDM), the mean variance over the subjects of this group is 3.021×10^{-2}, about 16 times larger than that of the second group (One-Class MDM > One-Class WG), 1.891×10^{-3}. This suggests that One-Class WG tends to perform better when the point cloud is more anisotropic. Although this difference is partly driven by a single outlier with unusually high variance, removing it still leaves the mean variance of the One-Class WG group 2.8 times larger. Statistical tests would strengthen this analysis, but with only 6 subjects in the second group, they would lack the power to detect significant differences.

6.7 Conclusion

In this chapter, we focused on a specific BCI dataset, that was recorded in a real surgical environment, and we were able to show that the wrapped Gaussian defined on the manifold $\mathcal{P}_d$ of SPD matrices could be useful to better differentiate the `awake` state from the `anesthetized`. Thus, a model that could look simple at first glance, the wrapped Gaussian, can still be interesting for real-life problems. This gives us an answer to our final question 3. The proposed approach significantly outperformed the baseline *One-Class MDM* with a clinically-relevant number of EEG sensors, with a lower inter-subject variability, highlighting the importance of modeling the dispersion on the Riemannian manifold rather than relying solely on distance to a centroid. Notably, an optimal performance was achieved with fewer electrodes, making the approach particularly suitable for a practical BCI deployment in surgical environments.

We mitigated the challenges associated with estimating the parameters of the wrapped Gaussian distribution discussed in Section 3.3, particularly in the low-sample regime, through several key modeling choices. Specifically, we assumed a wrapped Gaussian centered at $\mu = 0$ in the tangent space and restricted the dispersion matrix Σ to be diagonal. In addition, we reduced the number of electrodes used to compute the covariance matrices, thereby decreasing the complexity of the estimation problem. A more complex model would maybe lead to better results; however, we would need more training data in order to show it. Nevertheless, this shows that the wrapped Gaussian can still be useful when the data is limited. We showed that the improved results were mainly due to the anisotropy of the wrapped Gaussian. This is an important argument in the answering of question 3: the full wrapped Gaussian is maybe not the best model to use in practice, however, once several modeling choices are made according to the application or the amount of data available, this model can still be beneficial. The flexibility of the wrapped Gaussian allows it to be used in very different scenarios, even in very limited ones such as the one studied here.

Several directions may extend this work.

- First, the clear separation between awake and anesthetized scores observed in Figure 6.7 suggests that Riemannian change-point detection algorithms (X. Wang, R. A. Borsoi, et al. 2024; X. Wang, R. Borsoi, et al. 2025) could exploit the temporal structure of the data instead of treating trials independently. The goal of such change-point detection algorithms is to identify when the distribution of a time series changes. In our context, the consecutive covariance matrices can be seen as a time series where the points are the covariance matrices. As with the one-class problem we tackled in this chapter, one could see if using the anisotropy of the wrapped Gaussian could lead to better Riemannian change-point detection algorithms.
- In this chapter, we only test our algorithms on a single dataset, and we only compare our proposed One-Class WG to the standard One-Class MDM. However, in order to be adopted, our One-Class WG should be confronted to other One-Class Classifiers (Ruff et al. 2021) and validated on more datasets.
- More specifically, it would be great to have access to a dataset where a real AAGA is recorded. This is not the case in this study. However, Valérie Marissens Cueva has been collecting a new dataset with standard depth-of-anesthesia monitors, thus, such analyses will be possible in the near future.

Finally, a key aspect for clinical adoption is to study the interpretability of a given classifier, and more specifically in our context of Riemannian classifiers. Indeed, the Riemannian approach cannot be linked directly to what is known from a neurophysiological perspective, mainly because the covariance matrix is not known to be interpretable from the start. Building interpretability tools for Riemannian algorithms would help the BCI community adopt the Riemannian approach with more trust. This will be the goal of Chapter 7.

Chapter 7

Interpretability of Riemannian tools

Contents

7.1 Introduction 147
7.2 The Riemannian t-SNE 151
7.2.1 A Riemannian version of the t-SNE 151
7.2.2 Speeding-up the implementation 154
7.2.3 Numerical experiments 158
7.2.4 Using the t-SNE to better understand the data 163
7.3 Contribution of each electrode 164
7.3.1 Removing an electrode to assess its contribution 165
7.3.2 Permutation Feature Importance 166
7.4 Extension to other interpretability methods 169
7.4.1 Shapley Values 169
7.4.2 Counterfactual Explanations and Adversarial Attacks 170
7.5 Conclusion 171

7.1 Introduction

Having a model performing well is good, but understanding why it performs well is better. Knowing why a model works and how it takes its decisions builds trust, helps fix errors and can facilitate the evaluation of a system's fairness. This is the goal of *interpretable machine learning*: turning a black box into a white box. In Biran and Cotton (2017), later used by Molnar (2025), the authors define interpretability as "[...] the degree to which a human can understand the cause of a decision". The more a model is interpretable, the easier it is for a human to understand why certain decisions were made. In Brain-Computer Interfaces (BCI), the stakes of understanding the decision of an ML model are very high: from a medical point of view, we do not want to purely rely on a machine to take a decision. We still trust a human more than a computer, and thus, having a human capable of understanding the decision taken by an algorithm and validating it will always be very precious. Let us list some of the main stakeholders of a BCI system:

- There is the creator, a BCI researcher that builds and trains the system.
- Once the model is built, the experimenter will use the BCI on different subjects.
- Potentially, there are doctors, that use the BCI system to take decisions, as in Chapter 6 during general anesthesia. They might also train it (provided a simple enough software) for new patients.

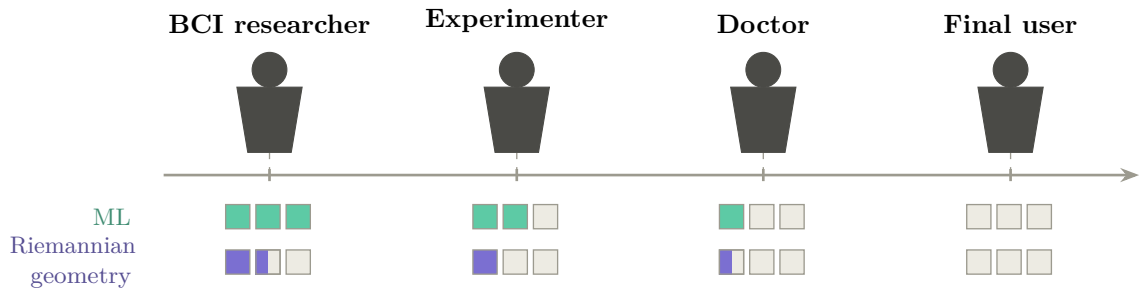


Figure 7.1. Spectrum of BCI stakeholders, from the most to the least knowledgeable in machine learning (ML) and Riemannian geometry. The BCI researcher masters ML and has at least some notions of Riemannian geometry, the experimenter has a working knowledge of ML but little Riemannian background, the doctor may have basic ML knowledge but little to no Riemannian background, and the final user is assumed to have neither.

- Of course, there is the final user, the person who actually controls the BCI, who has some feedback on its action and may improve their control over the system.

These different players have certainly different knowledge, probably different goals when using the BCI and need potentially different kinds of justification. On one end of the spectrum, the BCI researcher usually has some knowledge of machine learning (ML), knows how an ML pipeline is built, and can have little to good knowledge of Riemannian geometry applied to ML. The experimenter will have some BCI knowledge, basic understanding of ML, but maybe not of Riemannian geometry. The doctor might have some basic knowledge of ML but not necessarily, and it is acceptable to assume that they have very limited knowledge of Riemannian geometry. On the other end of the spectrum, we must assume that the final BCI user has neither knowledge of ML nor of Riemannian geometry. The final user can be anyone; however, they might ask for some understanding of the pipeline, and it is essential that BCI researchers give them some explanations and intuition on the system they are controlling. This spectrum of knowledge is illustrated in Figure 7.1. Thus, we must build tools helping the different stakeholders of BCIs to better apprehend, trust and understand Riemannian-based BCIs.

Moreover, for BCI, there are a number of possible biases and confounding factors that could be exploited by machine learning, but that would make the resulting BCI invalid. Indeed, the classifier's goal is only to optimize accuracy (or another metric) on a given training dataset, and it has no knowledge of what makes a "neurophysiological signal" coherent or not. For example, muscular or ocular artifacts (Fatourechi et al. 2007) can help the classifier reach a decision (Trocellier, N'Kaoua, et al. 2024), without relying on the true neural signal. Being able to detect this using interpretability tools, or ensuring that it is not the case, will help us build more trustworthy BCIs.

We should add that interpretability methods often require expert knowledge, as there is no ground truth for these methods. In order to assess that the results are coherent with what is known on the specific application domain, in our case neurophysiological knowledge, we need the help of an expert, who has prior knowledge of the specific application. They will be able to tell us if the output of an interpretability method is likely or not. Moreover, we are aware that the notion of interpretability is also criticized, such as by Lipton (2018), including the fact that post-hoc interpretations — interpretations done after the training of the model — can potentially be misleading.

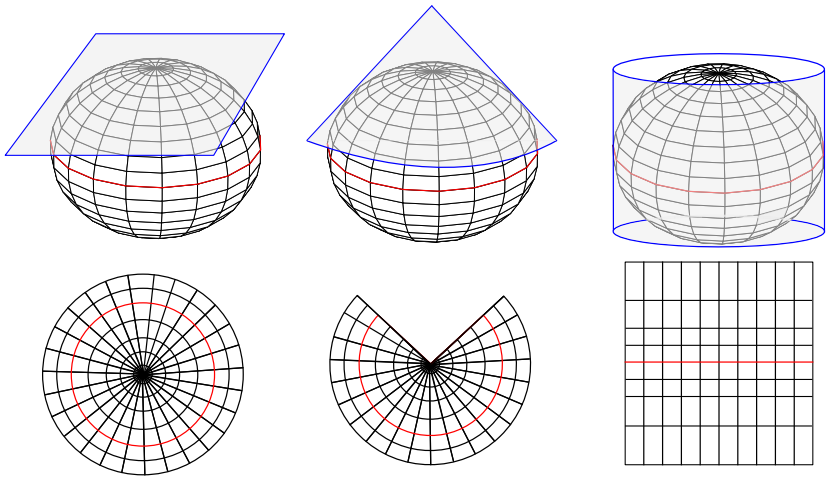


Figure 7.2. Different examples of projections of the curved two-dimensional surface of a globe on a plane. All three of these projections distort in one way or another the surface.

Visualizing high-dimensional SPD datasets

The main particularity of the ML pipelines that are studied in this thesis is that they take as input covariance matrices, or more generally SPD matrices, and that they respect the Riemannian geometry of the set $\mathcal{P}_d$ of SPD matrices. While Riemannian geometry is taught in every high-level mathematics curriculum, it is rare to find a non-mathematician knowing about Riemannian geometry. Our first goal in this chapter will be to build a tool helping people unfamiliar with Riemannian geometry to have a grasp on this geometry. In order to better apprehend this curved geometry, the first tool we will focus on is a visualization tool. For the final user, a visualization tool could be used as feedback, in order to better understand how their mental state is understood by the Riemannian classifier. Moreover, as stated by Unwin (2020), "Data visualization is useful for data cleaning, exploring data structure, detecting outliers and unusual groups, identifying trends and clusters, spotting local patterns, evaluating modeling output, and presenting results." Thus, for the BCI researcher and the doctor, such a tool can be used for a first high-level identification of specific structure in the data. As we stated in Section 2.1.3, BCIs are subject to numerous variabilities, both inter-subject and intra-subject. We hope that using a visualization tool, we will be able to better apprehend those variabilities.

Moreover, we recall that when an EEG is recorded using d electrodes, the covariance matrix estimated from the EEG will be of dimension $d(d+1)/2$. The SPD datasets we manipulate in practice are thus often high-dimensional. Unfortunately, as soon as one works with a point cloud of covariance matrices of dimension more than two, it becomes impossible to visualize the whole point cloud in a two or three-dimensional space. A common way to overcome this issue is to use a *dimensionality reduction* algorithm. The goal of such algorithms is to transform data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation contains some meaningful properties of the original data. The ideal case is when we reduce sufficiently the dimension so that we are able to visualize the low-dimensional representation (for example, the low-dimensional space is $\mathbb{R}^2$ or $\mathbb{R}^3$).

A major constraint when wanting to build a dimension reduction tool for the manifold $\mathcal{P}_d$ of SPD matrices is that this space is curved. Indeed, in this work, we mainly focus on the AIRM geometry of $\mathcal{P}_d$, described in Section 2.2.3.1. Using this geometry, the manifold $\mathcal{P}_d$ of SPD matrices is not flat, contrary to usual Euclidean spaces. This curvature implies that one cannot accurately reproduce a point cloud sampled in a Riemannian space into a Euclidean one. For example, it is well known (Mulcahy and Clarke 2001) that one cannot project the Earth (the curved two-dimensional Riemannian surface of a sphere) onto a plane without

distorting the surface in some way and to some extent (see Figure 7.2 for some examples of projection). The same phenomenon appears when dealing with SPD matrices. Thus, we need to build a specific dimension reduction algorithm from the manifold $\mathcal{P}_d$. In this chapter, we will adapt the *t-SNE* algorithm from Maaten and Hinton (2008) to the Riemannian manifold $\mathcal{P}_d$.

Going back to neurophysiological knowledge

The first tool we developed in this chapter is a Riemannian version of the t-SNE algorithm that gives us a better understanding of the datasets composed of SPD matrices. The dataset is the first part of the Riemannian ML pipeline. In the second part of this chapter, we will focus on interpretability methods for Riemannian classifiers. These interpretability methods are mostly built for BCI researchers and doctors, as a way of understanding the output of a Riemannian classifier. Indeed, a doctor, trained with a neurophysiological background, will certainly want to make sure that the decision made by the BCI is really based on some known neurophysiological features. Old-school ML methods for BCI, often built using hand-crafted features (see Section 2.1), are in general interpretable. In the foundational work on interpretability in BCI, Haufe et al. (2014), the authors make a clear distinction between the *forward model*, a model explaining how the data was generated from the neural sources, and the *backward model*, trying to reverse the data generation process. They also show that interpreting the backward model can lead to wrong conclusions, and develop a method to transform backward models into forward models. This method can for example be used to interpret the spatial filters obtained from the *Common Spatial Pattern* (see Section 2.1.1 or Ramoser et al. 2000; Blankertz, Tomioka, et al. 2008). A topomap is then usually used to visualize and interpret the results.

Unfortunately, the Riemannian approach cannot be linked directly to what is known from a neurophysiological perspective, mainly because the covariance matrix is not known to be interpretable from the start. Thus, we will seek interpretability methods built specifically for Riemannian pipelines, especially focused on linking a decision to the corresponding electrodes. For example, when performing a motor imagery task distinguishing the left hand from the right one, we know from the sensory Homunculus of Figure 2.3 that the area of the brain controlling the hands is the primary motor cortex. Thus, in a classical montage as the one given in Figure 2.2, the electrodes C3 and C4, or close ones, should be the most relevant for this task. Such interpretability tools can also help us discover some "clever Hans" effects, where the classifier is not taking the decision based on the right reason (Kauffmann et al. 2025).

Outline of the chapter

The first section of this chapter (Section 7.2) will be focused on adapting the t-SNE algorithm to the Riemannian setting. We will start by describing the Riemannian version of this algorithm in Section 7.2.1, then we will describe some numerical tricks to speed up the implementation of this Riemannian t-SNE in Section 7.2.2. Finally, we will conduct some numerical experiments to assess the quality of this algorithm in Section 7.2.3. The second section (Section 7.3), will be focused on assessing the contribution of the different electrodes to the classification result. For this, we will focus on two different methods in Section 7.3.1 and Section 7.3.2. Finally, we will see how other interpretability tools could be extended to Riemannian interpretability methods in Section 7.4.

The work done in Section 7.2 is based on the following paper, published in the journal *Transactions on Machine Learning Research*:

> **Thibault de Surrel**, Sylvain Chevallier, Fabien Lotte, and Florian Yger (2025). "Geometry-Aware visualization of high dimensional Symmetric Positive Definite matrices". In: *Transactions on Machine Learning Research*

Here, we propose an improved version, especially from the numerical point of view, with a significant speed-up of the algorithm based on mathematical and optimization tricks.

The second section is based on the following paper, published at MLSP 2025:

> **Thibault de Surrel**, Tristan Venot, Marie-Constance Corsi, and Florian Yger (2025). "Interpretability of Riemannian tools used in Brain Computer Interfaces". In: *2025 IEEE 35th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, pp. 1–6

We also propose a fresh perspective on this work, with new methods and extensions. Part of this fresh perspective was done with the help of our intern: Océane Mille.

7.2 The Riemannian t-SNE

The goal of this section is to adapt the Euclidean t-SNE algorithm from Maaten and Hinton (2008) to the Riemannian manifold $\mathcal{P}_d$ of SPD matrices. The final goal is to visualize a set of high-dimensional $d \times d$ SPD matrices while still taking into account the geometry of $\mathcal{P}_d$. For this, we need to find an output space that can easily be visualized while respecting the curvature of $\mathcal{P}_d$.

Luckily, the space $\mathcal{P}_2$ of 2×2 SPD matrices is a 3-dimensional manifold and can thus be plotted in a 3D space. Indeed, the space $\mathcal{P}_2$ can be rewritten as

$$\mathcal{P}_2 = \left\{ \begin{pmatrix} a & b \\ b & c \end{pmatrix} \mid a > 0, c > 0, ac > b^2 \right\}.$$

Therefore, one can represent in $\mathbb{R}^3$ a 2×2 SPD matrix using (a, b, c), and it will lie strictly inside the cone of equation $b = \pm\sqrt{ac}$ where $a, c > 0$. We give a representation of this cone in Figure 7.3.

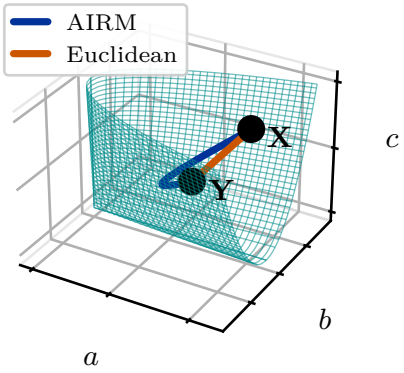


Figure 7.3. The set of 2×2 SPD matrices. The blue path is the AIRM geodesic between two SPD matrices and the orange path is the usual Euclidean path.

Thus, the set $\mathcal{P}_2$ of 2×2 SPD matrices will be the output space of our Riemannian t-SNE. The algorithm will seek an embedding of a high-dimensional point cloud of SPD matrices into a point cloud in $\mathcal{P}_2$, that will be then plotted in 3D. We want to build a qualitative embedding that preserves the local structure of the high-dimensional point cloud. This embedding will be solely used for visualization purposes. The t-SNE algorithm from Maaten and Hinton (2008) is thus perfectly suited for this task, as it is a similarity-based dimension-reduction technique widely used in data visualization (Wattenberg et al. 2016; Kobak and Berens 2019).

7.2.1 A Riemannian version of the t-SNE

Let us describe the Riemannian version of the t-SNE algorithm. Given a set of N high-dimensional data points $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ in $\mathcal{P}_d$, the goal of the t-SNE algorithm is to find $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_N \in \mathcal{P}_2$, a set of N 2×2 SPD matrices that resemble the high-dimensional points. The t-SNE algorithm tries to preserve local structure so that the low dimensional neighborhood

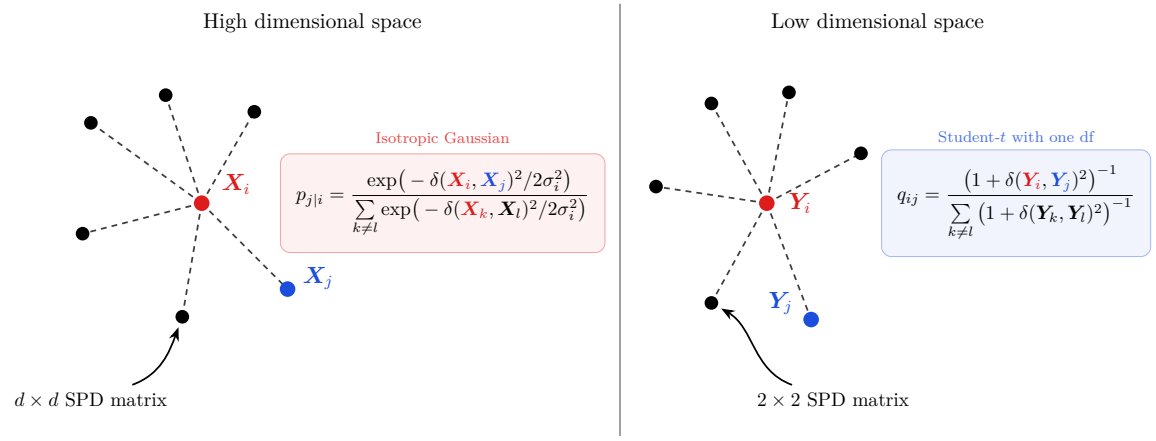


Figure 7.4. Illustration of the Riemannian t-SNE algorithm. In the high-dimensional space, the similarities are computed by centering an isotropic Gaussian on each point and computing the probabilities of the other points under this distribution. For the low-dimensional space, a Riemannian version of a Student t-distribution with one degree of freedom is used.

of each point is the same as the original neighborhood. The idea behind the t-SNE is to encode the high-dimensional data, and the neighborhood information, into a distribution. This is done by computing probabilities that represent similarities. The t-SNE algorithm is illustrated in Figure 7.4. In the Euclidean setting, the similarity of $\boldsymbol{X}_j$ to $\boldsymbol{X}_i$ is the conditional probability $p_{j|i}$ that $\boldsymbol{X}_i$ would pick $\boldsymbol{X}_j$ as its neighbor if they were picked in proportion to their probability density under a Gaussian centered at $\boldsymbol{X}_i$. To generalize this algorithm to the manifold $\mathcal{P}_d$ of SPD matrices, we use the Riemannian isotropic Gaussian from Said, Bombrun, et al. (2017) and already introduced in Chapter 2 and Chapter 4. Thus, the conditional probability $p_{j|i}$ is given by:

$$p_{j|i} = \frac{\exp(-\delta(\boldsymbol{X}_i, \boldsymbol{X}_j)^2/2\sigma_i^2)}{\sum_{k\neq l}\exp(-\delta(\boldsymbol{X}_k, \boldsymbol{X}_l)^2/2\sigma_i^2)} \tag{7.1}$$

where $\sigma_i > 0$ is the scaling of the Gaussian centered at $\boldsymbol{X}_i$. This parameter sets the size of the neighborhood around $\boldsymbol{X}_i$. Roughly speaking, the lower σ_i is, the smaller the neighborhood is, and thus all the weight will be on the nearest neighbor, and the higher σ_i is, the more uniform on all the points the weights are. As in the original t-SNE, σ_i is set differently for each high-dimensional point. It is set in such a way that the Shannon entropy from Shannon (1948) of the conditional distribution $P_i = (p_{j|i})_{j=1,\dots,N}$ equals a predefined Shannon entropy using a binary search. This entropy is controlled by the user through the perplexity parameter `perp`. Then, σ_i is computed such that

$$\texttt{perp} = 2^{H(P_i)},$$

where $H(P_i)$ is the Shannon entropy of P_i. We then symmetrize the conditional probabilities by setting $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2N}$ when $i \neq j$ and $p_{ii} = 0$. Apart from using the Riemannian isotropic Gaussian from Said, Bombrun, et al. (2017), this procedure does not differ from the original one introduced in Maaten and Hinton (2008).

Then, the t-SNE algorithm aims at learning $\boldsymbol{Y}_1, \dots, \boldsymbol{Y}_N \in \mathcal{P}_2$, that reflect the similarities p_{ij} as well as possible. For this, the Euclidean setting defines low dimensional affinities q_{ij} between $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$ based on a Student t-distribution with one degree of freedom. Compared to the Gaussian distribution, the Student t-distribution has a heavier tail. This way, dissimilar data points in high dimension will be modeled far away in low dimension while high-dimensional close points will stay close to each other. This is illustrated in Figure 7.5. We will extend this

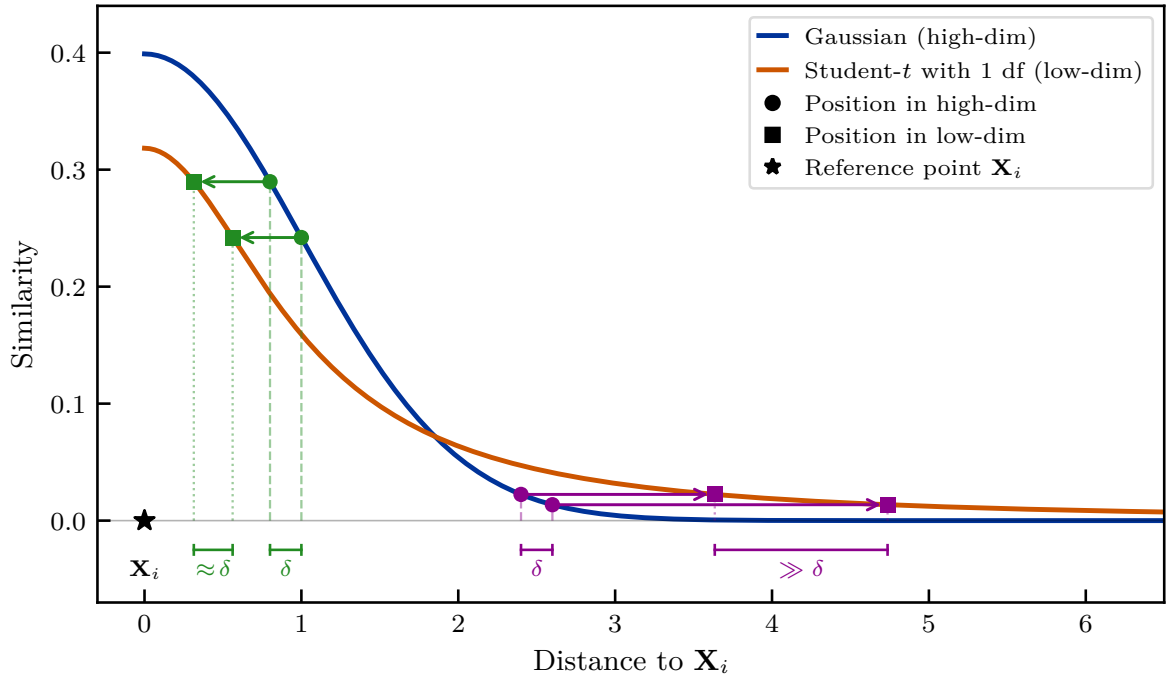


Figure 7.5. Illustration of the effect of choosing a heavy-tail Student t-distribution with one degree of freedom for the low dimensional affinities in the t-SNE. When computing the similarities to the reference point $\boldsymbol{X}_i$, the high-dimensional green circles are close to $\boldsymbol{X}_i$, so their low-dimensional green square representations are close to each other. On the contrary, the high-dimensional purple circles are far from $\boldsymbol{X}_i$, so their low-dimensional purple square representations are even further from each other.

distribution to the manifold of SPD matrices and define the joint probabilities q_{ij} as:

$$q_{ij} = \frac{(1 + \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2)^{-1}}{\sum_{k \neq l}(1 + \delta(\boldsymbol{Y}_k, \boldsymbol{Y}_l)^2)^{-1}}. \tag{7.2}$$

We then want to minimize the mismatch between the two conditional probabilities p_{ij} and q_{ij}. To measure this, the Riemannian t-SNE (as the Euclidean t-SNE) uses the Kullback-Leibler (KL) divergence and tries to minimize it with respect to the points $\boldsymbol{Y}_i$. The cost function C is then:

$$C = \sum_{i=1}^{N} \mathrm{KL}(P_i || Q_i) = \sum_{i=1}^{N} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}. \tag{7.3}$$

The minimization of the Kullback–Leibler divergence with respect to the points $\boldsymbol{Y}_i$ is performed using Riemannian gradient descent (Boumal 2023) on the product manifold $(\mathcal{P}_2)^N$. The result of this optimization is a set of low-dimensional points (2×2 SPD matrices) that reflects the similarities between the high-dimensional inputs. In order to perform this optimization, we need to compute the Riemannian gradient of the above cost defined in Equation (D.5).

Proposition 7.1 (Riemannian gradient of the t-SNE cost)**.** *The Riemannian gradient of the t-SNE cost defined as Equation* (D.5) *is given by:*

$$\nabla_{\boldsymbol{Y}_i} C = -4 \sum_{j=1}^{N} \frac{p_{ij} - q_{ij}}{1 + \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2} \operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j).$$

Proof. This gradient is very similar to the Euclidean one given in Equation 5 of Maaten and Hinton (2008). The main difference is that the gradient of the squared AIRM distance

$\delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2$ with respect to $\boldsymbol{Y}_i$ is

$$\nabla_{\boldsymbol{Y}_i} \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2 = -2 \operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j). \tag{7.4}$$

The full derivation is given in Appendix A.4.1. □

Remark 7.2. The t-SNE, and consequently the Riemannian t-SNE proposed here, is inherently a *transductive* method: the low-dimensional embedding is optimized for a specific high-dimensional point cloud and does not directly provide an embedding function for unseen data. This can be problematic in streaming settings, such as online BCI experiments, where spatial covariance matrices estimated from EEG data arrive sequentially. One possible approach to adapting t-SNE to such a setting is to optimize the embedding of newly arriving points while keeping the embeddings of previously observed points fixed, as proposed in H. Ma and K. Chen (2024).

NEW

7.2.2 Speeding-up the implementation

Now that we have introduced the Riemannian t-SNE algorithm, let us dive into implementation details on how we coded a fast version of this Riemannian t-SNE. Indeed, if we want practitioners to use this Riemannian t-SNE to understand their data, the minimization of the cost defined in Equation (D.5) must be efficient. This minimization is performed using a Riemannian gradient descent as described in Section 4.3 of Boumal (2023), we thus have two ways of speeding up this process: the first one is to speed up each iteration individually, and the second one is to see how the whole gradient descent can be improved. We dive into these two specific branches below.

7.2.2.1 Speeding up each iteration

At each iteration of the gradient descent, the first step is to compute the low dimensional similarities q_{ij} using Equation (7.2). This computation relies on computing the Riemannian distances between all the points $(\boldsymbol{Y}_i)_i$, thus computing the distance matrix of the $(\boldsymbol{Y}_i)_i$. Then, the second step is to compute the Riemannian gradient as given in Proposition 7.1. This gradient also relies on the distances between the low-dimensional points (that have already been computed for the q_{ij}), and computing the Riemannian logarithms $\operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j)$ for $i \neq j$. We recall that the formula for the AIRM distance is given in Equation (2.4) and the formula for the Riemannian logarithm is given in Proposition 2.19. They both rely on computing the eigenvalues of the inputs. A naive implementation of this Riemannian t-SNE, relying only on the default operations provided by pyRiemann (Barachant, Barthélemy, et al. 2026) is very slow (see Table 7.2, this version takes more than three minutes to compute 50 gradient steps for 2,000 matrices[1]). However, one can make an interesting and important remark:

All the computations for the minimization of the t-SNE loss are made in $\mathcal{P}_2$, the set of 2×2 SPD matrices.

We can leverage this remark to speed up the computations. Indeed, when given a 2×2 matrix, one can compute a closed-form formula of its eigenvalues.

Proposition 7.3. *Let* $\boldsymbol{X} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ *a* 2×2 *matrix. Then, its (possibly complex) eigenvalues are:*

$$\lambda_+ = \frac{a + d + \sqrt{(a-d)^2 + 4bc}}{2} \qquad \lambda_- = \frac{a + d - \sqrt{(a-d)^2 + 4bc}}{2}.$$

[1] In this chapter, all the numerical comparisons were done on a MacBook M3 Pro with 36 GB of RAM.

Function	$t_{\text{closed-form}}$ (s)	$t_{\texttt{pyRiemann}}$ (s)	Speed-up	Max error
Distance matrix	$0.0482_{\pm 0.0133}$	$0.5110_{\pm 0.0126}$	$\times 10.6$	3.33×10^{-8}
AIRM logarithm	$5.39 \times 10^{-5}{}_{\pm 0.0133}$	$0.0012_{\pm 0.00001}$	$\times 22.7$	7.88×10^{-15}

Table 7.1. Comparison of the closed-form formula for the AIRM distance and the AIRM logarithm on 2×2 SPD matrices to the pyRiemann implementation. We tested on 2,000 points and times are averaged over 10 runs. Errors are absolute differences between the two results.

Proof. We denote λ_+ and λ_- the two (possibly complex) eigenvalues of $\boldsymbol{X}$. We have the following:

$$\lambda_+ + \lambda_- = \operatorname{tr}(\boldsymbol{X}) = a + d, \qquad \lambda_+ \lambda_- = \det(\boldsymbol{X}) = ad - bc.$$

Substituting $\lambda_+ = a + d - \lambda_-$ into the second relation gives:

$$\lambda_-^2 - (a+d)\lambda_- + (ad - bc) = 0.$$

The discriminant simplifies as $\Delta = (a+d)^2 - 4(ad-bc) = (a-d)^2 + 4bc$, and the quadratic formula yields:

$$\lambda_\pm = \frac{a + d \pm \sqrt{(a-d)^2 + 4bc}}{2}.$$

□

Moreover, we recall that one of the two formulas to compute the AIRM distance given in Equation (2.4) is:

$$\delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j) = \sqrt{\sum_{k=1}^{2} \log(\lambda_k)^2},$$

where the λ_i are the eigenvalues of $\boldsymbol{Y}_i^{-1}\boldsymbol{Y}_j$. In our 2×2 case, we are easily able to compute in closed form $\boldsymbol{Y}_i^{-1}\boldsymbol{Y}_j$ and then, one can use Proposition 7.3 to compute the eigenvalues.

Let us compare the computational gains in practice. For this, we sample 2,000 random matrices $\boldsymbol{Y}_i$ following a wrapped Gaussian $\mathrm{WG}(I_2; 0_3, I_3)$, and we compare, on 10 runs, the times it takes to compute the distance matrix of the $\boldsymbol{Y}_i$ using the previous closed-form formula for the eigenvalues and using the generic `pairwise_distance` of pyRiemann that relies on NumPy's eigenvalue extractor. The results are given in Table 7.1 and are very clear: using the closed-form formula enables a $10\times$ speed-up, without any compromise on the numerical accuracy.

The other bottleneck of the optimization of the Riemannian t-SNE is the AIRM logarithm, that relies not only on the eigenvalues, but on the full eigendecomposition as it computes a matrix log. However, in the same spirit as for the eigenvalues, we have a simpler formula in the case of 2×2 SPD matrices.

Proposition 7.4 (AIRM logarithm for 2×2 SPD matrices)**.** *Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be two 2×2 SPD matrices, then, the AIRM logarithm given in Proposition 2.19 can be computed as*

$$\operatorname{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = c_1 \boldsymbol{Y} + c_2 \boldsymbol{X},$$

where

$$c_1 = \frac{\log(\lambda_+) - \log(\lambda_-)}{\lambda_+ - \lambda_-}, \quad c_2 = \frac{\lambda_+ \log(\lambda_-) - \lambda_- \log(\lambda_+)}{\lambda_+ - \lambda_-},$$

where $\lambda_\pm$ are the eigenvalues of $\boldsymbol{X}^{-1}\boldsymbol{Y}$.

Proof. We recall that the AIRM logarithm is

$$\operatorname{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{X}^{1/2} \log(\boldsymbol{X}^{-1/2} \boldsymbol{Y} \boldsymbol{X}^{-1/2}) \boldsymbol{X}^{1/2}.$$

$t_{\text{closed-form}}$ (s)	$t_{\texttt{pyRiemann}}$ (s)	Speed-up
$9.151_{\pm 0.132}$	$194.807_{\pm 2.404}$	$\times 21.3$

Table 7.2. Comparison between the vanilla implementation of the Riemannian t-SNE using pyRiemann and the boosted implementation using the speed-ups described in Section 7.2.2. The t-SNEs were fitted on 2,000 points with 50 gradient descent iterations. Times are averaged over 5 runs.

Let us start by focusing on $\log(\boldsymbol{Z})$ where $\boldsymbol{Z}$ is a 2×2 SPD matrix. Using Sylvester's formula (see Formulas 6.1.35 and 6.1.36 of Horn and Johnson 1994), we have, since $\boldsymbol{Z}$ is diagonalizable:

$$\log(\boldsymbol{Z}) = c_1 \boldsymbol{Z} + c_2 I_2,$$

where

$$c_1 = \frac{\log(\lambda_+) - \log(\lambda_-)}{\lambda_+ - \lambda_-}, \quad c_2 = \frac{\lambda_+ \log(\lambda_-) - \lambda_- \log(\lambda_+)}{\lambda_+ - \lambda_-},$$

and $\lambda_\pm$ are the two eigenvalues of $\boldsymbol{Z}$. Then, by choosing $\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2}$ for $\boldsymbol{Z}$ (which has the same eigenvalues as $\boldsymbol{X}^{-1}\boldsymbol{Y}$):

$$\operatorname{Log}_{\boldsymbol{X}}(\boldsymbol{Y}) = \boldsymbol{X}^{1/2}\left(c_1 \boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2} + c_2 I_2\right)\boldsymbol{X}^{1/2} = c_1 \boldsymbol{Y} + c_2 \boldsymbol{X}.$$

□

Thus, the AIRM logarithm is really easy to compute, and together with Proposition 7.3, it removes the need to compute any eigendecomposition. We run a similar experiment to the one above, measuring the time it takes to compute 2,000 AIRM logarithms with our implementation, and compared it with pyRiemann's implementation. The results are given at the second row of Table 7.1, and we can see that the speed-up is even bigger than for the AIRM distance: up to a $22\times$ speed-up.

> *Remark* 7.5. One could ask if such an optimization is specific to the case of 2×2 SPD matrices, or if it could be applied to SPD matrices of all shapes. The Sylvester's formula used for the proof of Proposition 7.4 can be stated in the general setting of a $d \times d$ diagonalizable matrix $\boldsymbol{Z}$, and relies on the eigenvalues of $\boldsymbol{Z}$. However, finding the eigenvalues of $\boldsymbol{Z}$ means finding the roots of a d-degree polynomial. For $d = 2$, the formula is simple. For $d = 3, 4$, formulas exist but can be lengthy, and for $d \geqslant 5$, we know from Galois's theory that a closed-form solution does not exist. See Chapter XVI of Gozard (2009) or Rotman (1990) for more information on this.

We can now compare our full *vanilla* Riemannian t-SNE implementation that uses pyRiemann's function with a boosted Riemannian t-SNE implementation using the tricks above. For this, we sample 2,000 random matrices $\boldsymbol{Y}_i$ following a wrapped Gaussian $\mathrm{WG}(I_2; 0_3, I_3)$, and we see how long both t-SNEs take to compute a low-dimensional embedding. We fix the number of iterations of the gradient descent to 50 such that the total fitting time is not too long, and we repeat this experiment 5 times. The results are given in Table 7.2, and one can see that the boosted version outperforms the vanilla implementation. We must emphasize that computing the high-dimensional similarities takes around 5 seconds for both algorithms (this part is implemented in the same way as we are dealing with high-dimensional matrices). Thus, the gradient descent for the pyRiemann version takes more than 3 minutes, while the boosted version takes only around 4 seconds, resulting in a $47\times$ speed-up. We also implemented a version of the boosted Riemannian t-SNE algorithm using Numba (Lam et al. 2015), a JIT compiler that translates Python and NumPy code into fast machine code. This version was around 30% faster. This is the version that will be used in the following.

7.2.2.2 Improving the whole minimization process

Now that each iteration of the Riemannian gradient descent is blazing fast, we can focus on optimizing the whole gradient descent. The speeding up of the Euclidean t-SNE has led to several works in the past 15 years (Maaten 2014; Linderman et al. 2019; Poličar et al. 2024; Häkkinen et al. 2020; Chan et al. 2018). While we will review some of the main ones in Remark 7.6, we will mainly focus on the classical tricks to improve the gradient descent of the t-SNE. We can first look at what the loss function looks like for the Riemannian t-SNE. We give a typical example of this loss in Figure 7.6. One can see that after a few hundred iterations, the loss does not evolve anymore.

This has also been observed for the Euclidean t-SNE and the typical number of iterations is set to 1000 (see the original implementation of Maaten and Hinton 2008, or the one provided by Scikit-learn for example). We will follow this heuristic and, in all the experiments, set the maximum number of iterations to 1000. Thus, in practice, we do not necessarily wait for the Riemannian gradient descent to converge, but we did not see any major degradation when doing this.

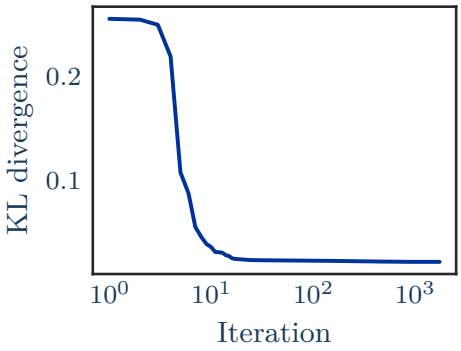


Figure 7.6. A typical illustration of the Riemannian t-SNE loss.

Another trick to improve the final embedding qualitatively introduced in the original Euclidean t-SNE is what Maaten and Hinton (2008) call *early exaggeration*. This trick is to multiply all the high-dimensional similarities p_{ij} by a constant factor α, in the original paper this factor is $\alpha = 4$, at the early phase of the optimization (typically the first 250 iterations). This forces the algorithm to find a good global structure before fine-tuning local structure. The early exaggeration procedure creates a landscape where clusters are well-separated early on, then focuses on the layout within each cluster. We empirically saw that having an early exaggeration of $\alpha = 2$ for the first 250 iterations helped improve the final embedding. Then, Belkina et al. (2019) and Kobak and Berens (2019) showed that a principled choice for the learning rate η is $\eta = N/\alpha$, where N is the number of points in the dataset and α is the early exaggeration factor. In our experiments, we will set the learning rate to be $N/2$, as we set the early exaggeration factor to $\alpha = 2$.

A last improvement for the Riemannian t-SNE is the choice of the initial point for the minimization. In the original implementation of the t-SNE, the low-dimensional points $\boldsymbol{Y}_i$ were initialized randomly, using a concentrated Gaussian. Belkina et al. (2019) and Kobak and Berens (2019) showed initializing t-SNE with the first two principal components, rather than random noise, results in embeddings that preserve global structure far more faithfully. To adapt this to the Riemannian setting, we will initialize the $\boldsymbol{Y}_i$ with a *Principal Geodesic Analysis* (P. Fletcher et al. 2004, PGA). For this, we compute the Fréchet mean $\hat{\boldsymbol{X}}_N$ of the high-dimensional dataset, we whiten the points: $\hat{\boldsymbol{X}}_N^{-1/2}\boldsymbol{X}_i\hat{\boldsymbol{X}}_N^{-1/2}$ and send them to the tangent space $T_{I_d}\mathcal{P}_d$ at identity using the Riemannian logarithm. In $T_{I_d}\mathcal{P}_d$, we perform a PCA, keeping only the two principal components. This gives us 2×2 symmetric matrices that we can send back to the manifold $\mathcal{P}_d$ using Exp_{I_2}. This set of points is used to initialize the Riemannian t-SNE.

> *Remark* 7.6. As mentioned at the beginning of this section, a lot of works have been proposed to speed up the t-SNE when the number of points N grows. A first follow-up to the classical t-SNE in Maaten (2014) uses a Barnes-Hut algorithm (J. Barnes and Hut 1986), an approximation algorithm for performing an N-body simulation. Using this algorithm to approximate the gradient of the t-SNE leads to an $\mathcal{O}(N \log N)$ algorithms, substantially accelerating the t-SNE. Later, Linderman et al. (2019) proposed FIt-SNE, which further

Dataset	Application	Dimension	# matrices
BNCI2014001	BCI	22×22	288×9
BNCI2014002	BCI	15×15	160×12
BNCI2014004	BCI	3×3	720×9
Air Quality	Atmospheric data	6×6	102
Textile	Image analysis	10×10	$4,000$
FPHA	Video sequences of hand actions	63×63	$1,175$

Table 7.3. Summary of the datasets used for the experiments comparing the dimension reduction algorithms.

reduces the computational complexity to $\mathcal{O}(N)$ by reformulating the repulsive gradient as a convolution, approximating it via interpolation onto an equispaced grid and computing it efficiently using the Fast Fourier Transform. Then, a GPU-accelerated version for the t-SNE arrived (Chan et al. 2018), using CUDA and exploiting the natural parallelism of the N-body force computation. t-SNE-CUDA achieves $50 - 700\times$ speed-ups over CPU implementations, enabling visualization of datasets like ImageNet (Deng et al. 2009) that were previously computationally intractable. Rather than focusing on the cost per iteration, in Häkkinen et al. (2020), the authors replaced the first-order momentum-based gradient descent with a quasi-Newton optimizer, achieving quadratic convergence and drastically reducing the number of iterations needed to converge. Adapting all of those variants of the original t-SNE to the Riemannian case could improve the algorithm proposed here and make it work on very large datasets.

7.2.3 Numerical experiments

We now want to validate our Riemannian t-SNE in numerical experiments. For this, we will conduct both qualitative and quantitative experiments on real-life datasets. For the following experiments, we will compare three versions of the t-SNE:

1. The **Riemannian t-SNE** that we propose in Section 7.2: the input and output distances are computed using the AIRM.
2. The **Riemannian-Euclidean t-SNE**: the input distances are computed using the AIRM, but the output are points from $\mathbb{R}^3$ equipped with the Euclidean structure.
3. The **Euclidean t-SNE**: the input distances are computed using the Euclidean structure of symmetric matrices based on the Frobenius norm, and the output are also Euclidean points in $\mathbb{R}^3$.

For the second and third version, we will use the Euclidean implementation of the t-SNE provided by Scikit-Learn (Pedregosa et al. 2011). For the perplexity parameter of our Riemannian t-SNE, we found empirically that using $\texttt{perp} = \frac{3}{4}N$, where N is the number of points, leads to the best results. The effect of the perplexity on the Riemannian t-SNE is empirically studied in Appendix C.1. When using the t-SNE from Scikit-Learn, we will use its default value: $\texttt{perp} = 30$. We will also compare our Riemannian t-SNE to other dimension reduction algorithms described in Section 2.3: the UMAP (McInnes et al. 2018), the Riemannian PCA (Horev et al. 2016) and the Principal Geodesic Analysis (PGA) (P. Fletcher et al. 2004). The output space of the UMAP and the PGA is $\mathbb{R}^3$ and for the Riemannian PCA, it is also $\mathcal{P}_2$. The input of the UMAP will be the Riemannian distance matrix.

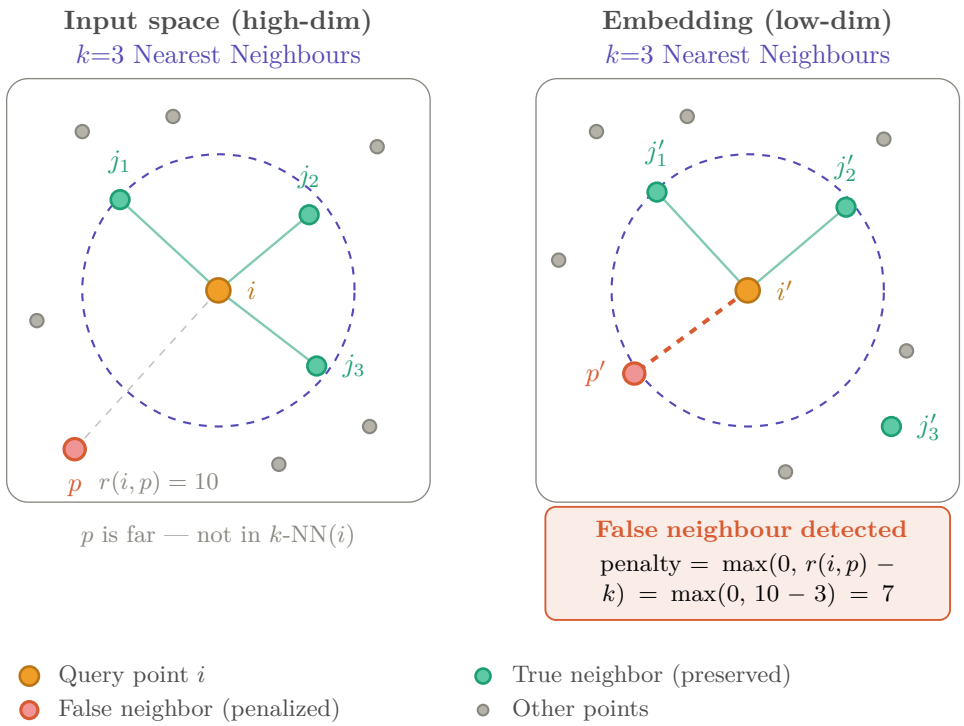


Figure 7.7. Illustration of how the *Trustworthiness* metric (Venna and Kaski 2001) is computed. On the left panel, we have the high-dimensional space. For a given query point i, we compute its true neighbors (here for $k = 3$ nearest neighbors). The point p is i's 10th nearest neighbor so $r(i, p) = 10$. Then in the right panel, representing the low-dimensional space, we see that a false neighbor (here, p') is amongst the $k = 3$ nearest neighbors. We compute its corresponding penalty: $\max(0, r(i, p) - k) = \max(0, 10 - 3) = 7$. We repeat this for all the query points, and we get the trustworthiness.

7.2.3.1 Setup

For this experiment, we will consider 6 different datasets from different applications. Most of the datasets that we will use have already been introduced in Chapter 4, and we summarize them in Table 7.3. The major differences are that we only consider $4,000$ random matrices in the dataset *Textile* over a total of $16,000$ in order to speed up the computations and that we introduce a new dataset: *FPHA*. This is a dataset of video sequences in first person view used as benchmark for estimating hand actions. The original dataset (Garcia-Hernando et al. 2018) consists of 1,175 video sequences distributed across 45 distinct categories. As in Rui Wang et al. (2022), we converted the video sequences into 63×63 SPD matrices representing how the 3D coordinates of 21 hand joints are linked together.

7.2.3.2 Metrics

The first metric that we will consider is: how it looks. Indeed, this is very important when wanting to interpret some datasets. The first metric will thus be a qualitative metric, and we will focus more on the BCI datasets to see if we are able to get some interesting insights about the data from the visualization. Then, from a quantitative point of view, we will use a metric called the *trustworthiness coefficient* (Venna and Kaski 2001). This metric indicates to what extent the local structure is retained. It ranges from 0 to 1 (with 1 being the highest score) and is computed by ranking the order of each point's neighbors and comparing these ranks between the original high-dimensional space and the reduced low-dimensional space. This coefficient depends on the number k of nearest neighbors one considers, and is computed

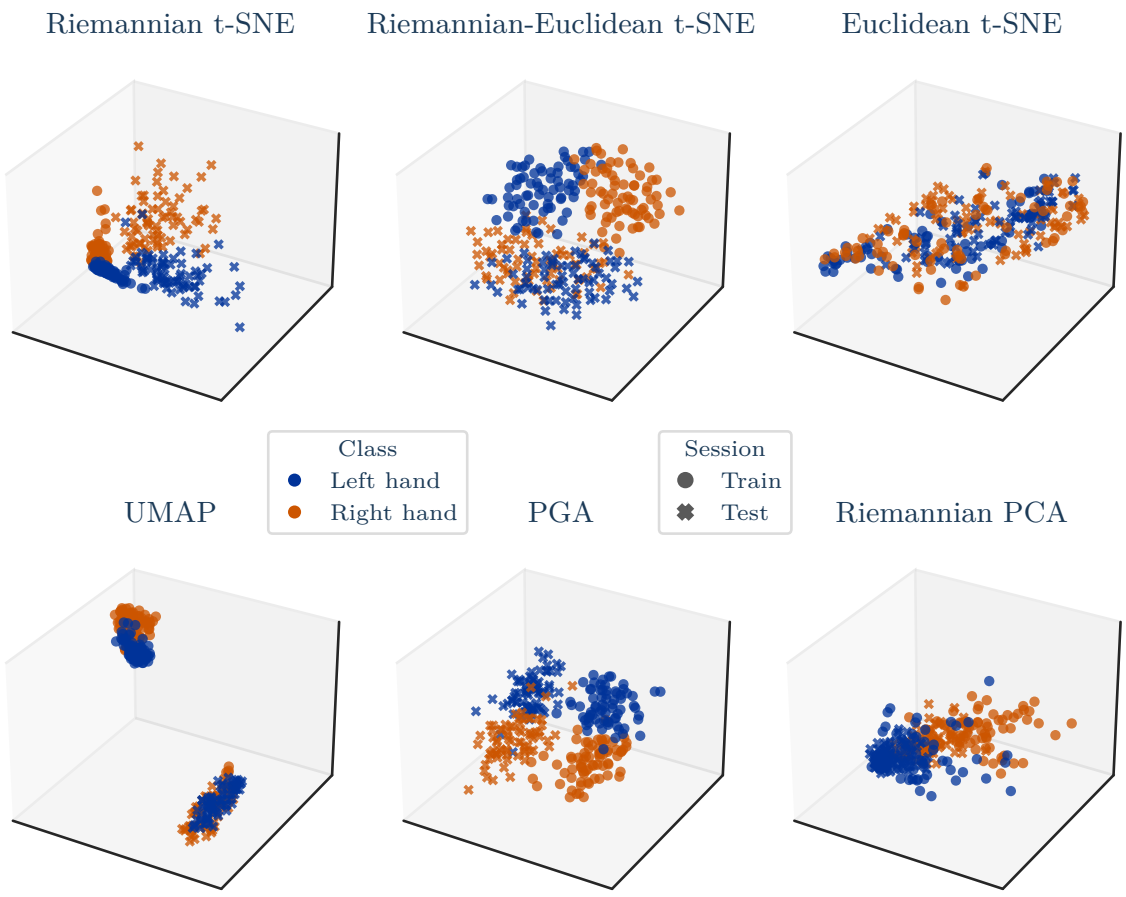


Figure 7.8. The different dimension reduction algorithms considered applied to Subject 8 of BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). Each point represents a covariance matrix. The colors correspond to the class and the shape of the markers correspond to the session.

as follows:

$$T(k) = 1 - \frac{2}{Nk(2N - 3k - 1)} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_i^k} \max(0, r(i,j) - k),$$

where for each sample i, $\mathcal{N}_i^k$ are its k nearest neighbors in the low-dimensional output space, and every sample j is its $r(i,j)$-th nearest neighbor in the high-dimensional input space. In other words, unexpected nearest neighbors in the output space are penalized according to their rank in the input space. This metric is illustrated in Figure 7.7. The number of nearest neighbors k is a hyperparameter that will help us understand to what extent the local and global structures of the point cloud are preserved. One needs $k < N/2$ where N is the total number of points.

7.2.3.3 Qualitative results

For the qualitative results, we have focused on a BCI dataset, the dataset BNCI2014001 from Tangermann, K.-R. Müller, et al. (2012). This dataset is interesting as it has several subjects performing a motor imagery task: they were asked to think about moving their right hand, or to move their left hand without moving it. Then, the EEG was recorded for 4 seconds. This dataset was recorded during two sessions: one called *Train* and the second one called *Test*. We used the different dimension reduction algorithms listed above on two specific subjects

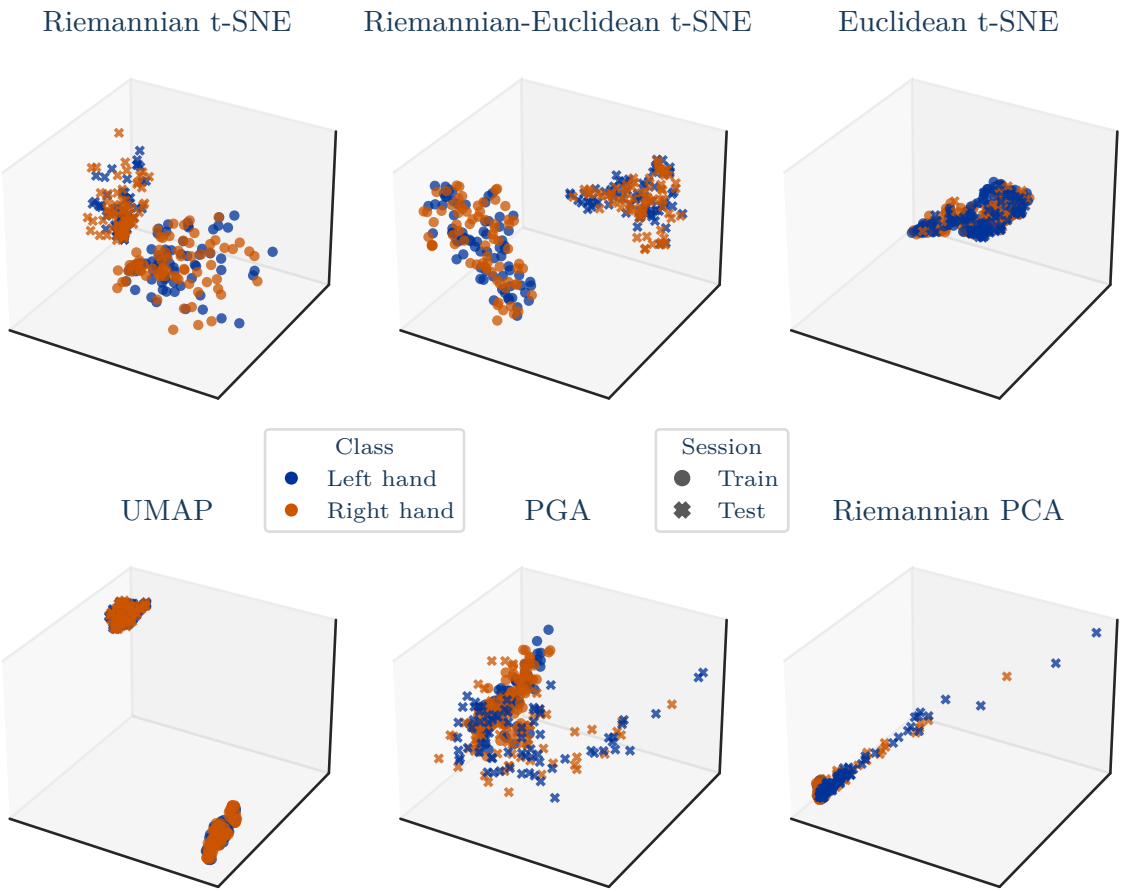


Figure 7.9. The different dimension reduction algorithms considered applied to Subject 2 of BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). Each point represents a covariance matrix. The colors correspond to the class and the shape of the markers correspond to the session.

of this dataset: *Subject 8* and *Subject 2*. We know that *Subject 8* has a good classification score when using a Riemannian classifier, whereas *Subject 2* has a low classification score. Thus, we expect the classes of *Subject 8* to be well separated while the classes of *Subject 2* are expected to be mixed. The different embeddings for *Subject 8* are given in Figure 7.8 while the embedding for *Subject 2* are given in Figure 7.9.

When looking at the results for *Subject 8*, one can see that the geometry-aware algorithms separate the two classes well (corresponding to the colors of the markers) while the Euclidean t-SNE completely mixes up the classes. As we said above, this subject has a high classification accuracy, so it is coherent to see such a separation between the classes. Moreover, we also see a separation between the sessions (the shape of the markers). This is not surprising as the signals recorded by an EEG during a BCI experiment are sensitive to the user's mental state (Saha and Baumert 2020), so it is likely that the data has shifted from one session to another. For *Subject 2*, the classes are completely mixed up, but the sessions are also well separated. From a quantitative point of view, we are able to see the SPD cone in the embeddings produced by the Riemannian t-SNE. This is coherent as, apart from the Riemannian PCA, it is the only algorithm that produces real SPD matrices, the other algorithms only produce 3D vectors.

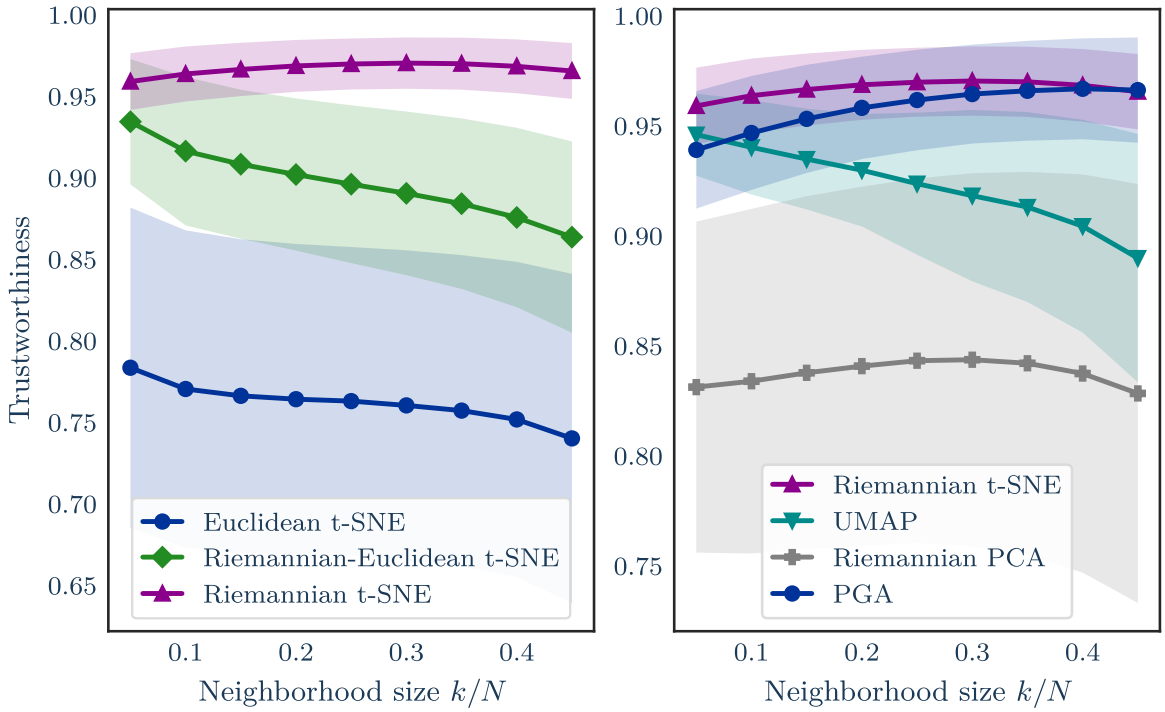


Figure 7.10. The results of the experiments on the real datasets: we plot the average of the trustworthiness coefficients over 6 datasets. On the left, we compare the different version of the t-SNE with each other. On the right, we compare the proposed Riemannian t-SNE with other dimension reduction tools. The x-axis represents the size of neighborhood used to compute the trustworthiness coefficient as % of the total number of points. The higher the trustworthiness is, the better.

7.2.3.4 Quantitative results

Our goal is also to quantitatively see how the local and global structures of the high-dimensional point cloud are preserved. For this, we look at how the trustworthiness coefficient evolves when the neighborhood expands. We give in Figure 7.10 the results of this experiment. We recall that the higher the trustworthiness coefficient is, the better the structure of the high-dimensional point cloud is preserved. A first important observation is that taking into account the Riemannian geometry of SPD matrices is important as the t-SNE works better when the high-dimensional distances are computed using the AIRM distance. When focusing on the different t-SNE, we see that the local structure (less than 10% of all the neighbors) is well-preserved using the Riemannian-Euclidean t-SNE but, as soon as the neighborhood gets bigger, the fully Riemannian t-SNE better preserves the structure. This is explainable as, locally, a Riemannian manifold can be approximated by a Euclidean space (the tangent space) but globally, the curvature induces some deformations of the space. This is exactly what we observe and therefore, the Riemannian t-SNE should be used when trying to reduce a dataset of SPD matrices.

When comparing our algorithms with PGA, UMAP and the Riemannian PCA, we clearly see that the Riemannian t-SNE is the best algorithm on these datasets. As UMAP does not have any *a priori* information on the geometry of the point cloud, it only manages to reproduce correctly the local structure of the point cloud and collapses as the neighborhood grows. The best competitor of our Riemannian t-SNE is the PGA, that seems to be able to preserve the global structure of the point cloud. However, our Riemannian t-SNE is better when it comes to local structure. One must also emphasize the fact that the PGA gives the principal components in the tangent space, thus Euclidean vectors, while our Riemannian t-SNE's outputs are SPD matrices. The Riemannian PCA greatly under-performs compared

to the proposed algorithms. All in all, the Riemannian t-SNE is the best trade-off between preserving the local and the global structures of a set of high-dimensional SPD matrices.

7.2.4 Using the t-SNE to better understand the data

In this first section, we have built a visualization tool in order to better apprehend point cloud of SPD matrices. Given a high-dimensional point cloud of SPD matrices, we compute a three-dimensional embedding of this point cloud, where the local structure is preserved as best as possible. In this manuscript, we are only able to plot two-dimensional rendering of three-dimensional plots, such as the ones proposed in Figure 7.8 and Figure 7.9. However, it is possible to build digital interfaces in which the user can move and manipulate the 3D plot, simplifying the understanding of the plot.

We also provide in Figure 7.11 the embedding obtained using this Riemannian t-SNE on one subject (subject 24) of the dataset used in Chapter 6. We recall that this dataset was recorded before and during general anesthesia with 64 electrodes. More details on this dataset are provided in Section 6.2.3. On this plot, the color corresponds to the dose of propofol injected to the subject. We can clearly see a separation between the `awake` state (when the concentration of propofol is zero), and the `anesthetized` state. This shows a clear separation between the mental states, which is captured by the *One-Class WG* proposed in Section 6.3.

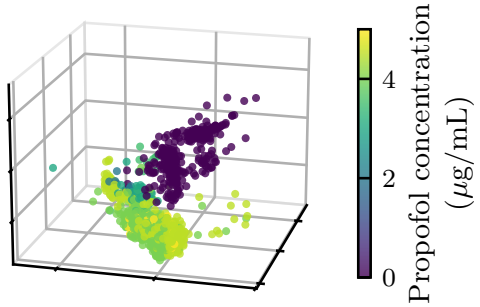


Figure 7.11. Riemannian t-SNE embedding of subject 24 of the dataset used in Chapter 6

As we saw in Figure 7.11 and with the qualitative results on a BCI dataset given in Section 7.2.3.3, we are clearly able to visualize the different mental states or sources of variabilities that exist in BCI in our plots. This tool can be useful in the first step of the Riemannian pipeline, in order to have a better intuition of the dataset we have in hand. In order to understand these plots, very little knowledge is required, thus they can be used by any stakeholder of a BCI system. We can use them to detect outliers, unexpected trends in the data or to validate clusters known *a priori*.

These plots are mostly qualitative, and even if we have shown that the trustworthiness coefficient is high when using the Riemannian t-SNE, we would recommend mainly using these plots to make qualitative measures. Indeed, after the t-SNE embedding, there is an important loss of dimensionality, required in order to visualize the data. For example, for the plots Figure 7.8 and Figure 7.9, the original dimension of the high-dimensional covariance matrices is $22 \times (22+1)/2 = 253$, and the dimension of the data plotted is $2 \times (2+1)/2 = 3$. This means that there is a significant loss of information. The work done here (except for the first part of Section 7.2.2), can be extended to compute an embedding where the SPD matrices are of any size. However, the t-SNE is mainly a tool used to visualize the data (Maaten and Hinton 2008; Wattenberg et al. 2016; Kobak and Berens 2019), and not as a pre-processing tool to reduce the dimension of the data, unlike PCA for example. Thus, we only consider a 2×2 embedding, the only one we are able to visualize.

7.3 Contribution of each electrode

In the first part of this chapter, we built a data visualization tool that takes into account the geometry of SPD matrices. This tool can help visualize a set of covariance matrices from one or several subjects, thus helping us better understand real high-dimensional datasets and better apprehend the Riemannian geometry of $\mathcal{P}_d$. This new tool, the Riemannian t-SNE, is to be used on the raw data, before being fed to any classifier. In this second part, the goal is to better understand and interpret the result of a Riemannian classifier, and see if the output of such classifiers can be linked to what is known from a neurophysiological perspective.

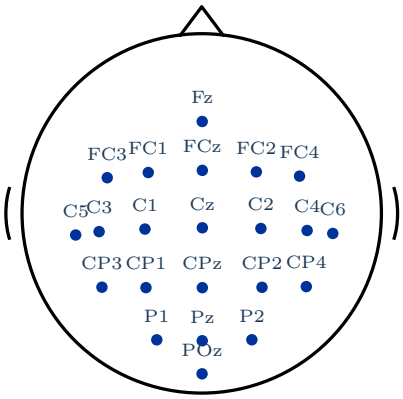


Figure 7.12. The EEG montage used in BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012)

In order to do this, we will focus on the influence of each electrode in the final classification result. Using the sensory Homunculus given in Figure 2.3, we roughly know which zone of the brain controls which part of the body. Thus, we want to see if the "important" electrodes for the Riemannian classification are correlated with the correct zones of the brain. This will give us a first interpretation to establish a link with the sensorimotor areas involved in motor imagery tasks and will provide a first step toward completely understanding the output of a Riemannian classifier. To assess the importance of each electrode, we will use two different methods: the first one will be to remove the electrodes and compare the classification results before and after removal and the second one will be to use permutation feature importance (Breiman 2001). We will then discuss how to extend this work.

These different interpretability methods will be tested on the dataset BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). This dataset is composed of 9 subjects, performing a motor imagery task and was recorded with 22 electrodes (see Figure 7.12). While this dataset is composed of 4 classes (`left_hand`, `right_hand`, `feet` and `tongue`), in order to facilitate the interpretation, we will only focus on 2 classes: `left_hand` and `right_hand`. From the sensory Homunculus of Figure 2.3, we know that the area of the brain controlling the hands is part of the primary motor cortex. Given a standard EEG montage, such as the one displayed in Figure 7.12, the main electrodes covering this area are C3 and C4. Event-related desynchronization (ERD) in the α (8–13 Hz) and β (14–30 Hz) bands over these electrodes is the classic signature used to decode hand motor imagery (G. Pfurtscheller and Lopes da Silva 1999). Moreover, we will divide the subjects into two groups: subjects with a "good" classification score ($\geqslant 75\%$) and subjects with a "bad" classification score ($< 75\%$). We will mainly focus on the mean results using the different techniques for those groups using the *Minimum Distance to Mean* (MDM) classifier described in Section 4.3.1 or in Barachant, Bonnet, et al. (2011). However, we will provide in Appendix C.2 plots for each subject individually using different classifiers.

Using a similar taxonomy as the one provided by Molnar (2025), we will differentiate between *local methods* and *global methods*. The goal of a *local method* is to explain **individual predictions**: they provide a "zoomed-in" view of a given model through the lens of a specific instance or trial. On the contrary, a *global method* aims at describing how features affect prediction on average, across a dataset. Moreover, almost all the interpretability methods described here will be *model agnostic post-hoc methods*, meaning that they can be applied to any type of classifier, even if we will only focus on Riemannian classifiers, and that they are meant to be used after the model has been trained. The methods ignore what is inside the specific models, and only focus on the output of the given models, and how changing the

input affects the output.

For the different methods described below, we will start by describing the method itself, and then give the results and interpretation on the dataset BNCI2014001.

7.3.1 Removing an electrode to assess its contribution

7.3.1.1 The method

The first **global** method we used to measure the influence of an electrode on the Riemannian classification result is to remove it from the data and compare the classification accuracy before and after removing it. Let us denote $(\boldsymbol{X}_i)_{i=1,\dots,N}$ the set of covariance matrices estimated from the EEG (see Section 2.1.4.2). If the EEG was recorded using d electrodes, the covariances matrices will be of shape $d \times d$. We will refer to this dataset as the set of *full* covariance matrices. Now, for $j \in \{1, ..., d\}$, we denote by $(\boldsymbol{X}_i^{(-j)})_{i=1,\dots,N}$ the set of *partial* covariance matrices estimated after removing channel j. These matrices are therefore of dimension $(d-1) \times (d-1)$ and correspond to the set $(\boldsymbol{X}_i)_{i=1,\dots,N}$ where we have removed the j-th line and the j-th column.

Given a Riemannian classifier, one can train it using the set of *full* covariance matrices $(\boldsymbol{X}_i)_{i=1,\dots,N}$ or using the set of *partial* covariance matrices $(\boldsymbol{X}_i^{(-j)})_{i=1,\dots,N}$. We can then compare the score s^{full} of the *full* classifier and the score $s^{(-j)}$ for the *partial* classifier by computing their relative error:

$$\frac{s^{(-j)} - s^{\text{full}}}{s^{\text{full}}}.$$

As this gives us one value per sensor, we can plot all the relative errors using a topographic map. An important point of this tool is that it is *classifier agnostic*, meaning that one can plug any Riemannian classifier. We must however note that removing an electrode will not remove all of the signal of the sources located under this electrode: indeed, as the electrodes in a classical EEG setting are quite near to each other (see Figure 7.12) and that EEG has a quite low spatial resolution (see Section 2.1), the signal of a given source will be measured and recorded by several electrodes. This is a clear limit to our approach.

Using this method helps us identify useful electrodes: indeed, if removing an electrode does not change the accuracy of the classifier, then this electrode can be considered as useless. On the contrary, removing a very useful electrode will certainly degrade the classification performance. Moreover, if removing an electrode improves classification, it could be that this is a noisy or artifact electrode, which is also useful information. By plotting a topographic map of the relative errors for each electrode, we will have a visual representation of the useful and useless electrodes.

7.3.1.2 The results

In Figure 7.13, we apply this first method for the classifier MDM on the dataset BNCI2014-001 (Tangermann, K.-R. Müller, et al. 2012). We plot the mean topographic maps for the subjects with a classification score that is greater than 0.75 and for the subjects with a classification score that is less than 0.75. When the relative error is greater than zero (the areas in red in the plot), this means that removing the sensors improves the classifier whereas when the relative error is less than zero (the blue areas), this indicates that removing the sensors decreases the classification score. We can see that most of the sensors, for the subjects with a classification accuracy of over 75%, are in blue, meaning that they all bring useful information to the classifier and that removing them deteriorates the classification accuracy. Moreover, for the subjects with an accuracy that is less than 75%, most sensors are in red, meaning that they most likely bring some noise to the signal and removing them improves the classification. Nevertheless, for all subjects, the sensors C3 and C4 are very useful as they are in blue. This is

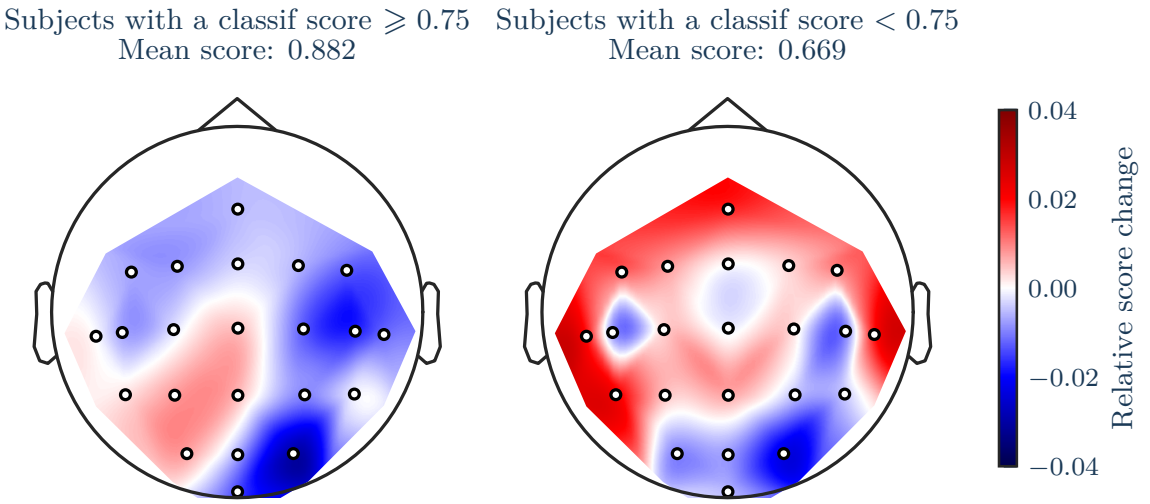


Figure 7.13. Mean of the influence of each electrode on the classification using an MDM. Dataset: BNCI2014-001 (Tangermann, K.-R. Müller, et al. 2012). The relative error between the score of an MDM trained with the "partial" covariance matrices and the "full" ones is plotted here for each electrode. The bluer the color, the more useful the electrodes are for the classification. The right plot corresponds to subjects with a classification score $\geqslant 0.75$ and the left one to the other subjects.

coherent as the motor imagery actions were `left_hand` versus `right_hand` and the sensors C3 and C4 are the ones that most capture this information. Therefore, the important information for the Riemannian point of view is the same as from a neurophysiological point of view. However, for the subjects with an accuracy over 75%, some sensors seem to be important from a Riemannian perspective even if it would not be the case neurophysiologically. For example, the sensors at the back of the head (P2, Pz, POz) are positioned over the visual cortex. Therefore, neurophysiologically, they would be less prone to provide relevant information regarding the classification of `left_hand` versus `right_hand`. However, from a Riemannian point of view, it seems that they bring useful information for the classification, as removing them worsens the results. A possible explanation is that the Riemannian classifier behaves as a "clever Hans" (Lapuschkin et al. 2019) and uses some signals triggered by the visual cue shown to the subject to indicate which action to mentally perform. Another possibility is that the BCI users are doing visual imagery (instead of kinesthetic motor imagery) or that they are paying visual attention left or right, thus involving the visual cortex (Christa Neuper et al. 2005).

7.3.2 Permutation Feature Importance

In the spirit of occlusion approaches for interpretability, the first *naive* method we introduced consists in removing the channel from one electrode. However, this necessitates the retraining of a new classifier on the smaller-dimension data. Thus, this amounts to measuring the gap between two distinct learning problems, one on $\mathcal{P}_d$ with $d \times d$ covariance matrices and one on $\mathcal{P}_{d-1}$, with $(d-1) \times (d-1)$ covariance matrices. When using the previous method, we are not really interested in interpreting the result of a given classifier, as we are comparing different classifiers trained on different data.

In order to improve the interpretability of a Riemannian classifier, we can look at known interpretability methods. In this section, we focus on *Permutation Feature Importance*, a **global** method introduced in Breiman (2001) for Random Forests, later extended to a model agnostic version by A. Fisher et al. (2019). We mainly refer to Chapter 23 of Molnar (2025) for a good introduction to Permutation Feature Importance.

7.3.2.1 The method

This *Permutation Feature Importance* method measures the importance of a feature (in our case, of an electrode) by looking at the increase in the prediction error of the model after a permutation of the values of this feature is done. This permutation breaks the relationship between the feature and the true outcome, helping us determine how much a model relies on this feature. When a feature is important for the classification, shuffling its values will decrease the classification accuracy.

In order to use this Permutation Feature Importance in the case of EEGs, we start by fitting a Riemannian classifier on 80% of the data. Then, given an electrode $j \in \{1, ..., d\}$, we randomly permute the j-th channel of the remaining 20% of testing trials between them. Thus, the j-th channel of a given testing EEG trial will be the j-th channel of another randomly selected testing EEG trial. Then, we can estimate the resulting covariance matrices and compute the classifier's error with these new covariance matrices. The Permutation Feature Importance score is then computed similarly as above, using the relative error between the score before permutation and the one after. 10 different permutations were used, along with 10 different splits, and the values were averaged into a single importance score for each sensor.

Remark 7.7. One could wonder about the effect of shuffling the time samples of the j-th channel of the EEG on the estimated covariance matrices. As we are shuffling the values of the j-th channel of a given EEG with the values of the j-th channel of another random EEG, we are breaking the dependencies and correlations between the other channels and channel j. Thus, when computing the covariance matrix of the permuted EEG, the j-th row and j-th column will have very small values compared to the original covariance matrix. This is what we observe in practice: we show in Figure 7.14 the mean covariance matrix we obtain after perturbing channel CP4 of subject 8 of BNCI2014001. The row and the column corresponding to this channel in the covariance matrix have very small values compared to the rest of the covariance matrix. Of course, the coefficient (j, j) corresponds to the variance of the j-th EEG channel, thus it depends only on the value of the j-th channel, and therefore, its value is not zero.

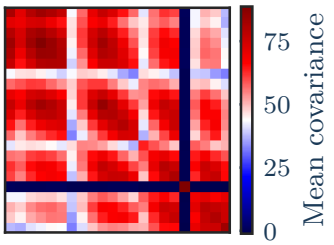


Figure 7.14. Mean covariance matrix estimated after permuting channel CP4 in the EEGs of patient 8 of BNCI2014001.

7.3.2.2 The results

We plot the Permutation Feature Importance score for the dataset BNCI2014001 and for the classifier MDM in Figure 7.15. For subjects achieving high classification accuracy, the result is even clearer than when we removed the electrodes. Here, we clearly see that the electrodes C3 and C4 play a very important role in the classification of `right_hand` versus `left_hand`. Indeed, when permuting those channels, the classification accuracy drops by more than 20%. The importance map reveals well localized discriminative regions, with the largest importance values concentrated on center electrodes on both hemispheres. The most influential channels are located around the sensorimotor cortex, which is consistent with the expected activation during right/left hand motor imagery. A slight asymmetry can also be observed, with channels over the right hemisphere exhibiting, on average, higher importance values than their left-hemisphere counterparts. We do not see the activation of the visual cortex using this method, as we did at Figure 7.13. If this area of the brain is really used as a clever Hans, this interpretability method is not able to detect it.

In contrast, subjects with lower classification accuracy exhibit uniformly low permutation

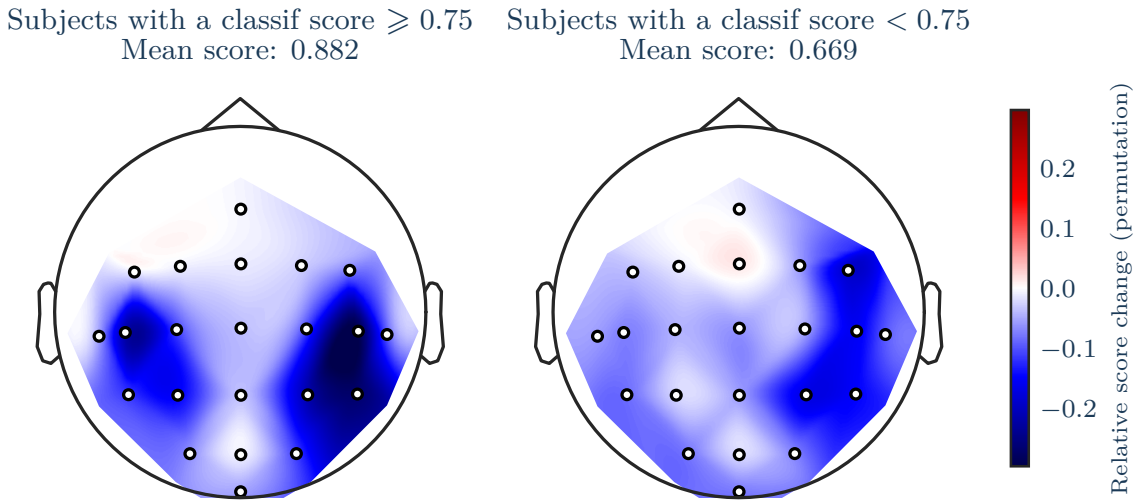


Figure 7.15. Mean result for the Permutation Feature Importance on the dataset BNCI2014001. For each channel, the values are randomly permuted across the trials, the covariance matrix is re-estimated and the classification score of this perturbed covariance matrix is compared to the original one. The bluer the color, the more the permutation of this channel affects the classification result.

feature importance across all channels. Rather than identifying a small set of highly informative sensors, the classifier appears to rely on weakly discriminative information distributed across the scalp. This result suggests that no channel provides a strong contribution to the decision process for these subjects, which is consistent with their lower decoding performance.

Extension to a fully Riemannian method In the Permutation Feature Importance method we presented above, one needs to have access to the original EEG in order to use it. Indeed, in this method, we permute the channels of the EEG, and then we estimate the covariance matrices of the permuted EEGs. Now, let us assume that we do not have access to the original EEGs, but only to the estimated covariance matrices. How could we adapt Permutation Feature Importance to this setup? Remark 7.7 gives us a possible answer. In fact, instead of permuting the channels in the EEG, we simply set the j-th row and the j-th column of the covariance matrix to zero, except for the diagonal element (j, j). This does not require having access to the full EEG, and simulates having randomly permuted the j-th channel with data that is not correlated to the rest of the EEG. As we are trying to interpret Riemannian classifiers, we need to make sure that the perturbed matrix we created is still SPD.

As we are keeping a strictly positive element on the diagonal (the variance of the j-th channel denoted σ_j^2), the perturbed matrix will always stay SPD. Indeed, up to a permutation (that does not change the SPDness of the matrix), the perturbed matrix can be assumed to be block-diagonal, with one block being $\sigma_j^2 > 0$ and the other block a $(d-1) \times (d-1)$ block. This second block is a principal sub-matrix of the original covariance matrix, thus it is SPD. Moreover, a block-diagonal matrix with SPD blocks is SPD. So this operation cannot break SPDness.

We implemented the fully Riemannian variant of the Permutation Feature Importance, and plot the results for the subjects achieving a high accuracy score in Figure 7.16. The results were very similar to the classical Permutation Feature Importance plotted at Figure 7.15. The C3 and C4

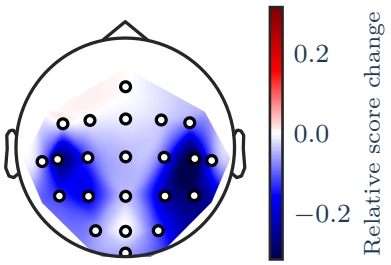


Figure 7.16. Fully Riemannian Permutation Feature Importance on the subjects performing well of BNCI2014001.

electrodes are also the ones with the highest importance. Quantitatively, the mean difference between the two ways of computing the Permutation Feature Importance ratios is 0.0051 and the maximum difference is 0.024. Thus, the fully Riemannian method can be used instead of the original Permutation Feature Importance method. It requires fewer computations as one only estimates the covariance matrices once.

7.4 Extension to other interpretability methods



At this point, we presented two methods to help interpret the result of a Riemannian classifier trained on covariance matrices from EEGs: a first method where a whole channel is removed to access its contribution to the classification result and another method using the Permutation Feature Importance method, where a channel is randomly permuted and the effect on the classification score is also measured. These are only the first steps in understanding completely how Riemannian classifiers take their decision, and if the taken decisions are based on known neurophysiological factors. However, a myriad of other interpretability methods have been developed for classical networks. We refer to the very complete book of Molnar (2025) for an introduction to these different methods. During the writing of this manuscript, we had an intern, Océane Mille, that worked on using and adapting interpretability tools for BCI Riemannian classifiers. These are the tools we started investigating.

7.4.1 Shapley Values

Compared to the two methods presented in Section 7.3.1 and Section 7.3.2, the Shapley Value is a **model agnostic local** method, which focuses on explaining individual predictions. The question asked by the Shapley Value is: given a specific point, how much has each feature value contributed to the prediction compared to the average prediction (Molnar 2025). Or, reformulated, given a prediction for one instance, how much did each feature contribute to pushing the prediction away from some baseline? The Shapley Value, introduced by Shapley (1953), comes from game theory and was suggested for machine learning interpretability by Štrumbelj and Kononenko (2011) and Štrumbelj and Kononenko (2014). In the context of machine learning, computing a Shapley value requires evaluating how the model's prediction for a given instance changes as different subsets of features, known as coalitions, are treated as known, with the remaining features treated as missing (Rozemberczki et al. 2022). However, trained classifiers usually require all features to produce a prediction. To counter this issue, classical libraries computing the Shapley value, such as `shap` (Lundberg and S.-I. Lee 2017), sample the missing features from their marginal distribution, i.e., replace the missing features with values from randomly drawn training instances and average the resulting predictions. One must then average each feature's marginal contribution over every possible ordering in which features could be revealed, weighted so that each coalition size is treated fairly. Since this is a difficult combinatorial problem, exact computation is generally intractable, and approximate methods are used instead: Štrumbelj and Kononenko (2014) propose a Monte-Carlo estimator that samples random feature orderings directly, while Lundberg and S.-I. Lee (2017) propose `KernelSHAP`, which recovers the same quantity by sampling coalitions and solving a weighted linear regression.

One could use the Shapley Values on EEGs, in order to assess the contribution of each electrode. However, this would necessitate the use of the EEG and the estimation of their covariance matrix at each step in order to use a Riemannian classifier. As we did for the Permutation Feature Importance in Section 7.3.2, we would like to get rid of this dependency on the EEGs, and have a fully Riemannian version of the Shapley values. This raises a major problem when computing the prediction of a Riemannian classifier after changing some of its features by randomly selected values from another trial, as it is done in the computation

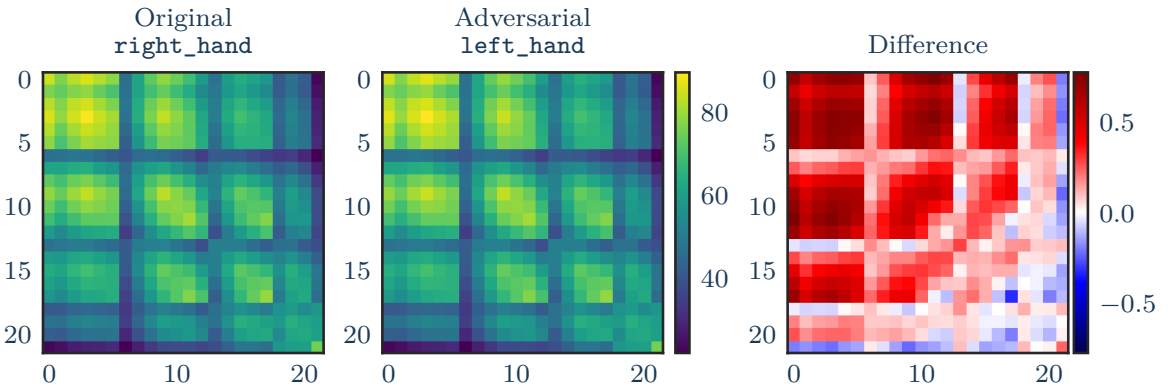


Figure 7.17. Example of a Riemannian adversarial attack on the dataset BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). The left covariance matrix is the original one, correctly classified as `right_hand` by the SPDNet (Huang and Van Gool 2017; Aristimunha, Ju, et al. 2026). The middle covariance matrix is the adversarial one, incorrectly classified as `left_hand` by the SPDNet. On the right, we plotted the difference between the true and the adversarial covariance matrices. One can see that the adversarial one is visually indistinguishable from the true one. Here, the budget of the attack was set to $\varepsilon = 0.1$. This Riemannian adversarial framework is taken from our work Timoz et al. (2026).

of the classical Shapley value. Indeed, doing this will most likely break the SPDness of the input matrix, and thus, the Riemannian classifier will not be able to work. In order to fix that, a solution would be to project the perturbed matrix onto the set $\mathcal{P}_d$ of SPD matrices, for example by replacing negative eigenvalues by a small $\varepsilon > 0$. However, there is maybe a better way of computing a fully Riemannian Shapley value, and this question remains open at the time of writing this manuscript.

7.4.2 Counterfactual Explanations and Adversarial Attacks

Another way of explaining the prediction of a given instance (a data point) is to build a *counterfactual explanation*. This leads to a **local method** and, according to Molnar (2025), the goal of such an explanation is to find "the smallest change to the feature values that changes the prediction to a predefined output". This will give a human-friendly explanation because it is contrastive to the current instance. When dealing with SPD matrices, we thus want to slightly modify the SPD matrix in order to change the prediction of the model. However, as our model is Riemannian, the perturbed matrix should stay SPD. A way of doing this would be to change a subset of coefficients in the SPD matrix (and its symmetric coefficient in order to keep the symmetry) in a way such that the new matrix is still positive-definite. Finding such a transformation is still under investigation at the time of writing this manuscript.

Another, more classical way of building a counterfactual explanation is *Adversarial Attacks* (Carlini et al. 2019; A. Chakraborty et al. 2021), which are particularly well suited when the classifier is a Neural Network. An adversarial attack is a way of finding a small perturbation of a given point that will fool the model. In our context, the neural network of interest is the *SPDNet* from Huang and Van Gool (2017) and described in Section 2.3.5. The specificity of this network is that it has been built to take as input SPD matrices and preserves the SPDness of the input at each layer (except the last one when doing classification). Thus, the goal of an adversarial attack is, given a correctly classified point $\boldsymbol{X} \in \mathcal{P}_d$, and an attack budget $\varepsilon > 0$, to find $\boldsymbol{X}_{\text{attack}}$ such that $\delta(\boldsymbol{X}, \boldsymbol{X}_{\text{attack}}) < \varepsilon$ and $\boldsymbol{X}_{\text{attack}}$ is misclassified by the model. A typical way of finding such adversarial examples is to optimize a point that maximizes the loss of the trained network while restricting the search space to a ball of radius

ε around $\boldsymbol{X}$. This can be done in a *white-box setting*, where one has access to the loss and the weights of the models. We studied a Riemannian way of doing a *Projected Gradient Ascent* (Madry et al. 2017) and applied it to SPDNets trained on EEG covariance matrices in Timoz et al. (2026). We show an example of adversarial attack on BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012) in Figure 7.17. We also showed how to go back to the EEGs, by crafting a perturbation of the EEG that still fools the SPDNet. When one does not have access to the weights of the model, nor to its loss function, the setup is called *black-box*. For SPDNet, we studied this setup in Cavarretta, **de Surrel**, et al. (2026). Here, we focused on the logits of the model and tried to find a point that maximizes the logit to the desired class. In order to solve this optimization problem, we relied on different derivative-free optimization algorithms that do not require the knowledge of the true gradients. Some algorithms estimate a directional derivative (Nesterov and Spokoiny 2017; H. Cai et al. 2021), others relied on Riemannian Direct Search (Kungurtsev et al. 2024; Cavarretta, Goyens, et al. 2026). We showed that black-box attacks were also capable of fooling a trained SPDNet. Other methods based on Riemannian optimization with finite-difference gradient approximations as explored in Taminiau et al. (2026) could also be explored in a similar setting.

Those adversarial examples do not give us a straightforward way to interpret the results of a machine learning pipeline, unlike the methods presented in Section 7.3.1 and Section 7.3.2. However, it is a step in the same direction, as adversarial attacks help us identify what important features need to be changed, and how to change them, in order to fool the classifier into another class. Moreover, if a tiny, imperceptible perturbation flips a classification, this suggests that the model's decision boundary is highly sensitive along certain directions, perhaps directions that humans do not perceive as meaningful (Ilyas et al. 2019). The budget used to build an adversarial attack also provides information on wether a point if far from the decision boundary or not, giving a proxy on the robustness of a model at a specific point. We have not yet studied the implications of the different Riemannian adversarial attacks we have built through the prism of interpretability, however, when the classifier of interest is an SPDNet, or any of its variant, it is certainly a promising path. Another major advantage of these Riemannian adversarial attacks compared to the other methods described above is that they are specifically designed to work on SPD matrices, and not on the EEG itself. Thus, they can be used in other applications outside BCI, as long as the inputs of the neural network of interest are SPD matrices.

7.5 Conclusion

The goal of this chapter was to give new tools to better understand and interpret the results of a Riemannian BCI pipeline. We proposed here different tools, that can be used by the different stakeholders of a BCI, according to their specific knowledge of machine learning or Riemannian geometry.

The first tool we propose, the Riemannian t-SNE, lets anybody visualize a high-dimensional point cloud of SPD matrices in 3D. This tool can help the BCI researcher or the doctor identify some trends or clusters, visualize the variabilities or the different mental states of a subject during an experiment. More generally, this visualization tool can also be used while developing new methods for dealing with SPD matrices, helping researchers better visualize the effect of a given transformation compared to another or how the dataset evolves through the learning pipeline. In order to improve the accessibility of this tool to the BCI research community (and more generally, the community of learners on $\mathcal{P}_d$), we added the Riemannian t-SNE to the Python libraries pyRiemann (Barachant, Barthélemy, et al. 2026). We got some feedback from different colleagues that already used our Riemannian t-SNE in order to better apprehend the datasets they were using.

In this work, we focused on an *a posteriori* use of the Riemannian t-SNE, meaning that

Method	Local or Global	Model Agnostic
Removing an electrode	Global	Yes
Permutation Feature Importance	Global	Yes
Shapley Value	Local	Yes
Counterfactual Explanations	Local	Yes*

* The special type of Counterfactual Explanations that are *Adversarial Attacks* are best suited for Neural Networks.

Table 7.4. Comparison of the interpretability methods presented in this chapter.

we visualize a set of SPD matrices once they have all been recorded. Our goal is now to see if this tool can also be used online, for feedback purposes. As we described in Section 2.1, the feedback step of a BCI is essential. It helps the user of the BCI to know if the system is understanding what they are inputting, and how to adapt in case the BCI fails. One could imagine that the user has a real-life visualization of their mental state using the Riemannian t-SNE. A target could also be added to this visualization, and the user could see how their different mental states can go closer to or further from this target. In order to develop such an online interface, we need to rethink how the optimization process is done (H. Ma and K. Chen 2024). Indeed, as new points will be added to the dataset to visualize gradually, maybe it is not necessary to re-fit the entire t-SNE when a new point arrives. One could rather optimize the position of these new points very quickly, while fixing the others and, every 10 to 15 new points added, re-optimize everything. We started investigating this new optimization procedure, without conducting a complete study yet.

In the second part of this chapter, we focused on interpretability methods for Riemannian classifiers summarized in Table 7.4. We started by showcasing two different methods to study the effect of the different electrodes on the classification result. The first method was based on removing an electrode from the recorded EEG, retraining a Riemannian classifier on this smaller dimensional data, and comparing the classification accuracy to the original accuracy. The second method relied on Permutation Feature Importance, where the values of a given channel are randomly permuted across the different trials, and the classification accuracy of the perturbed covariance matrix is compared to the original one. For both methods, we showed that the most important electrodes were neurophysiologically consistent. We also saw that the method using the Permutation Feature Importance led to clearer results, particularly for the subjects achieving a high accuracy, and the results were really about a given trained classifier, while the first method required the re-training of a classifier. Something that was important in this study is the possibility to use those methods without having access to the original EEG. Thus, we wanted to adapt these methods to a fully Riemannian setup. We showed that both methods can be derived in this setup, and that it was possible to use them when having access only to the covariance matrices and for any Riemannian classifier.

Lastly, we discuss other interpretability methods, in particular the Shapley value and counterfactual explanations. A lot of work still needs to be done to fully understand how Riemannian classifiers really take their decision, and those methods could be useful. For the Shapley value, the extension to a fully Riemannian setup is not trivial, while for counterfactual explanations, at least using adversarial attacks, we have already done some work paving the way.

This last chapter devoted to interpretability is a little bit apart from the others. The goal of this chapter is not really to answer any of the questions that were asked in Chapter 1. Nevertheless, the questions asked here are also essential, and must be asked in parallel to the rest. One must, of course, try to develop better statistical models for covariance matrices estimated from EEGs, better Riemannian classifiers based on these data; however, one must

also understand the decisions taken by an algorithm. If the hope is that, one day, these algorithms will be used in a real-life scenario, we must be sure that everyone using them trusts and understands (at least to some extent) these methods. Trustworthiness is essential when deploying algorithms in the real world, and this chapter tries to give some answers to better apprehend these obscure Riemannian methods.

8

Chapter 8

Discussion & perspectives

Contents

8.1 Summary of the thesis 175
8.2 Discussion 178
8.2.1 Taking a step back on the construction of the wrapped Gaussian . . 178
8.2.2 A word on the extrinsic methods 179
8.2.3 When the theory meets the applications 179
8.2.4 What really makes the wrapped Gaussian useful? 180
8.3 Limitations 181
8.3.1 Complexity of the model 181
8.3.2 Numerical complexity 182
8.3.3 Limitations of interpretability tools 182
8.4 Perspectives 182
8.4.1 Short-term ideas 183
8.4.2 Long-term ideas 185
8.5 Concluding Remarks 188

8.1 Summary of the thesis

Let us recall the main questions we asked in the introduction

> *Question* 1. What geometry-aware statistical models yield useful representations on the manifold of SPD matrices?

In this thesis, we chose to answer this question by building and studying a specific probability distribution on the manifold $\mathcal{P}_d$ of SPD matrices: the *wrapped Gaussian*. The study of this distribution is the main contribution of this work. As described in Chapter 3, this distribution depends on three parameters: $\boldsymbol{P} \in \mathcal{P}_d$, the anchor of the wrapped Gaussian that sets the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ on which we define the classical multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$, where $\mu \in \mathbb{R}^D$ will be the mean of this distribution in the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ and $\Sigma \in \mathcal{P}_D$, the dispersion matrix of the distribution. This dispersion matrix can be a full $D \times D$ matrix (with the condition of being SPD), thus it defines an anisotropic distribution: non-uniformity in different directions.

The main difference between the wrapped Gaussian we study and other wrapped distributions proposed in Galaz-Garcia et al. (2022), Nagano et al. (2019), and Troshin and Niculae

(2023) is that we allow the mean μ in the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ to be non-zero. We showed in Section 3.2.4 that allowing this leads to an identifiability issue: the full wrapped Gaussian model $\{\mathrm{WG}(\boldsymbol{P};\mu,\Sigma), \boldsymbol{P}\in\mathcal{P}_d, \mu\in\mathbb{R}^D, \text{ and } \Sigma\in\mathcal{P}_D\}$ is not identifiable. This is related to the existence of a "flat direction" in $\mathcal{P}_d$ (see Table 3.2): moving the parameters along this flat direction will not change the wrapped Gaussian. Several other theoretical properties of the wrapped Gaussian were studied, such as some of its moments or a wrapped version of the Central Limit Theorem.

An important part of question 1 is that we want our statistical model to be **useful**. A first step in showcasing its usefulness is to show that its parameters can be estimated. Indeed, if we are capable of showing that, given a sample from an unknown wrapped Gaussian, we are able to retrieve the true parameters, then this parametric distribution can be used to model real data, for which the underlying parameters are unknown. In Section 3.3, we tackled this estimation problem by studying the *Maximum Likelihood Estimator* for the wrapped Gaussian, first deriving some theoretical statements on the latter, and then by running some numerical experiments. We showed that we were capable of estimating the parameters of a wrapped Gaussian with an acceptable error. However, we showed that the number of samples required to have a good estimation was very high, and thus simplifying the structure of the model (with a diagonal dispersion matrix Σ), would help with this estimation problem.

The usefulness of the wrapped Gaussian can also be analyzed through its applicability for building learning tools for $\mathcal{P}_d$. This is what question 2, that we recall below, is about:

> *Question 2.* Can this statistical model help design better learning tools for SPD matrices?

We mainly answered this question in Chapter 4 and Chapter 5 from two different points of view: a parametric point of view and a non-parametric one. In Chapter 4, we showed that different learning tools, that were already known and used on $\mathcal{P}_d$, could be united in a general framework, built using two types of Riemannian Gaussians: the isotropic Gaussian from Said, Bombrun, et al. (2017) and the wrapped Gaussian from Chapter 3. Classifiers such as the MDM or the Tangent Space LDA/QDA from (Barachant, Bonnet, et al. 2011), which are classical Riemannian classification methods used in BCI, outlier detection tools such as the Riemannian potato from (Barachant, Andreev, et al. 2013; Barthélemy et al. 2019) or Riemannian kernels used on $\mathcal{P}_d$ can all be written in terms of isotropic Gaussians or wrapped Gaussians. When the underlying distribution is the isotropic Gaussian, we call these methods *intrinsic*, as they are defined directly on the manifold, without any approximation of the space. On the contrary, other methods rely on a local Euclidean approximation of the curved manifold $\mathcal{P}_d$, the tangent space. We call these methods *extrinsic*, and they are based on the wrapped Gaussian.

Another characteristic of the intrinsic methods relying on the isotropic Gaussian is that, as the name of the Gaussian suggests, they are *isotropic*, meaning that their spread is equal in every direction. For example, the MDM only captures first-order moments by estimating the Fréchet mean of each class, without investigating higher-order moments. However, real-life data is unlikely to be isotropic and the wrapped Gaussian allows us to model this anisotropy through the dispersion matrix Σ. Thanks to this, we were able in Chapter 4 to extend some of the isotropic tools to anisotropic ones by leveraging the wrapped Gaussian. This gave birth to the *Wrapped Discriminant Analysis* and to the *Wrapped Potato* that will become the *One-Class WG* in Chapter 6. We compared numerically the different classifiers in Section 4.3.5, showing, as in the estimation experiments in Section 3.3.3, that, provided with enough samples of each class, the Wrapped Discriminant Analysis algorithms could perform better than state-of-the-art methods.

In Chapter 4, we mainly focused on parametric tools, for which we assumed that the data would follow a parametric distribution (the isotropic Gaussian or the wrapped Gaussian). Then, in Chapter 5, we extended the probabilistic framework initialized in Chapter 4 to a

specific non-parametric method: *Flow Matching* (Lipman, R. T. Q. Chen, et al. 2023). This method allows us to sample from an unknown data distribution by learning a transformation from a Gaussian in the Euclidean case, a wrapped Gaussian in our Riemannian case, to the unknown data distribution. Using the wrapped Gaussian to build probability paths between the source and data distributions, we were able to propose a general framework for Conditional Flow Matching on $\mathcal{P}_d$, encompassing state-of-the-art methods such as RiemannianCFM from Z. Chen, Yue Song, G. Liu, et al. (2024) and DiffeoCFM from Collas, Ju, et al. (2026). Moreover, we proposed a new method based on this framework, the *WrappedCFM* that improves theoretically on both methods previously cited. In practice, we showed that our WrappedCFM produced better results than DiffeoCFM while being competitive with RiemannianCFM.

These chapters gave us a solid answer to question 2: indeed, the wrapped Gaussian is capable of building useful learning tools, provided enough training data. The tools we studied in the first part of this thesis were general tools, meant for all types of SPD matrices. However, we have an important application in mind: *Brain-Computer Interfaces* (BCI). The last question 3 is focused on this application:

> *Question 3.* Can one use this statistical model to better tackle variabilities observed in BCI?

This last question was mainly tackled in Chapter 6, where we focused on a specific application: detecting wakefulness during general anesthesia. The specificity of this application is that we do not have access to the two classes during training. Indeed, before the surgery starts, we only have access to EEGs during which the subject is `awake`; we do not have any samples of the opposite class `anesthetized`. Thus, we transform the Riemannian Potato from Barachant, Andreev, et al. (2013) described in Section 4.4 and the proposed *Wrapped Potato* into One-Class classifiers: the *One-Class MDM* and the *One-Class WG*. As we do not have a lot of training data available, we restrict the wrapped Gaussian to be centered in the tangent space, thus setting $\mu = 0$. This is only one of the restrictions we enforced on our wrapped Gaussian to alleviate data scarcity: we actually showed in Section 6.4 that our One-Class WG produced the best results when considering only a very low number of electrodes, 5 electrodes, and a diagonal dispersion matrix Σ. This setup is nevertheless very interesting for this specific application, as, during a real-life surgery, having a simpler setup with fewer electrodes will always be better.

We also conducted an analysis to better understand why the One-Class WG produced better results than the One-Class MDM. We showed that it was certainly due to the anisotropy of the wrapped Gaussian. This defining property makes the wrapped Gaussian more capable of handling real-life, complex datasets, making the algorithms more efficient. Thus, it gives an answer both to question 2 and to question 3: the wrapped Gaussian is capable of producing better learning tools by better handling the variabilities in a BCI dataset.

Finally, the last Chapter 7 is a departure from the questions we asked above. This last chapter does not rely on a Riemannian probability distribution, nor does it try to build a new learning tool. On the contrary, this chapter focuses on the interpretability of existing Riemannian tools. The goal here is to better understand the Riemannian geometry at play in BCI, to have a better intuition of a set of covariance matrices and to better apprehend the decisions taken by a Riemannian classifier. For this, we propose two types of tools. The first one is a visualization tool: in Section 7.2, we adapted the t-SNE algorithm from Maaten and Hinton (2008) to the manifold $\mathcal{P}_d$ of SPD matrices[1]. This tool allows us to visualize in three dimensions a high-dimensional set of SPD matrices, while still respecting the geometry of $\mathcal{P}_d$. Thanks to this tool, we are able to better understand the variabilities that occur in a BCI dataset (see Figure 7.8 and Figure 7.9). Then, we proposed different interpretability tools for Riemannian classifiers, in order to validate neurophysiologically the decisions taken by such classifiers. For this, we started by assessing the importance of each electrode by removing

[1]This Riemannian t-SNE is based on the isotropic Gaussian, so everything is still connected.

it and looking at the difference in terms of performance once removed. Then, we adapted another tool, the Permutation Feature Importance from Breiman (2001), that shuffles the values of a given electrode across the trials to assess its importance. Using both methods, we were able to show that the decisions taken by an MDM were aligned with neurophysiological knowledge. Finally, we discuss several extensions of different tools that could be used to interpret the results of a Riemannian classifier.

This last chapter is also an important part of machine learning that should not be neglected. When it comes to BCIs, the tools we build are meant to be used by people other than machine learning experts, such as doctors or the final BCI user. Being capable of explaining the decisions taken by an algorithm is crucial in building trust and easing its applicability.

8.2 Discussion

Now that we have summarized the different contributions made in this thesis, let us discuss some of its points. The goal here is to provide a more intuitive understanding of why things work as they should, at least from the author's perspective.

8.2.1 Taking a step back on the construction of the wrapped Gaussian

Let us first focus on the main object of this thesis, the *wrapped Gaussian*. This theoretical object is the one that drove nearly all the contributions made in this thesis, except from Chapter 7. Let us then argue intuitively and qualitatively why we studied this object.

When we started to work on probability distributions for the manifold $\mathcal{P}_d$ of SPD matrices, our main *desiderata* for this distribution was its **anisotropy**. Indeed, we were already aware of the isotropic Gaussian of Said, Bombrun, et al. (2017), and we had in mind that real-life data, such as spatial covariance matrices estimated from EEGs, was not isotropic. This is where the quest for an anisotropic Gaussian began. According to Wikipedia[2], "An anisotropic object or pattern has properties that differ according to direction of measurement". This means that an anisotropic distribution would have some preferred or privileged directions in space. In a Riemannian setup, how does one encode a direction? Using a tangent vector. Thus, an anisotropic Riemannian distribution must be characterized by a certain set of tangent vectors that will encode the preferred directions of the distribution.

Let us consider $\boldsymbol{U}_1, ..., \boldsymbol{U}_k$, k non-zero tangent vectors at a point $\boldsymbol{P} \in \mathcal{P}_d$. These will be the preferred directions of our anisotropic Riemannian Gaussian. A way to have a random direction that takes into account these preferred directions would be to consider the following random tangent vector $\boldsymbol{V}$:

$$\boldsymbol{V} = t_1\boldsymbol{U}_1 + ... + t_k\boldsymbol{U}_k \in T_{\boldsymbol{P}}\mathcal{P}_d, \qquad \text{where } t_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1). \tag{8.1}$$

Then, using the properties of the Euclidean Gaussian, one has that $\boldsymbol{V}$ is a random vector on $T_{\boldsymbol{P}}\mathcal{P}_d$ whose distribution is

$$\boldsymbol{V} \sim \mathcal{N}\left(0_D, \Sigma\right),$$

where Σ is a certain SPD matrix[3] that depends on the $\boldsymbol{U}_i$. In order to transform this random direction into a random point on the manifold $\mathcal{P}_d$, one can simply use the Riemannian exponential mapping at $\boldsymbol{P}$ by computing:

$$\boldsymbol{X} = \mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{V}).$$

[2] https://en.wikipedia.org/wiki/Anisotropy, accessed August 21st, 2026.

[3] Actually, Σ is SPD only under the assumption that the $\boldsymbol{U}_i$ span the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$.

Then, one has exactly that $\boldsymbol{X} \sim \mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$ from Definition D.1. Moreover, by allowing the t_i to be non-centered one-dimensional Gaussians (i.e. $t_i \sim \mathcal{N}(\mu_i, 1)$) in Equation (8.1), then one gets $\boldsymbol{V} \sim \mathcal{N}(\mu, \Sigma)$, where $\mu \in T_{\boldsymbol{P}}\mathcal{P}_d$ depends on the $\boldsymbol{U}_i$. Finally, this leads to $\boldsymbol{X} \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$.

This gives us a first intuition on why the wrapped Gaussian is a "good" distribution to study when looking for a Riemannian anisotropic distribution.

8.2.2 A word on the extrinsic methods

In Section 2.3.6, we state a clear difference between two types of methods, the *intrinsic* and the *extrinsic* methods. While the intrinsic methods are defined directly on the manifold and are most often specifically designed for specific curved spaces (MDM, AIRM kernel etc.), the extrinsic ones rely on a local linear approximation of the curved manifold using the tangent space. Then, one maps the data onto a well-chosen tangent space and applies a Euclidean algorithm in this tangent space (TS-LDA/QDA, log-Euclidean kernels etc.).

Extrinsic methods can be very appealing as the linear structure of the tangent space allows the user to use any off-the-shelf Euclidean learning method, simplifying the construction of new learning tools for Riemannian data. Moreover, if the algorithm produces any type of data, such as a regression process as in Sabbagh et al. (2019) and Mellot, Collas, S. Chevallier, Gramfort, et al. (2024), or dimension reduction algorithms (Sommer, Lauze, and M. Nielsen 2014; J. Liu et al. 2026), then one can simply map it back onto the manifold using the Riemannian exponential.

However, as explained in Section 7 of Jaquier and Rozo (2027), single tangent space approaches have different flaws. A first one is that mapping data from the manifold to the tangent space changes the "nature" of the points: they become *tangent vectors*, which can be interpreted, as we saw above, as *directions* or *velocities* on the manifold. Thus, extrinsic methods are actually classifying, regressing, or reducing the dimension of velocities instead of points on the manifold. For example, when doing Flow Matching as described in Chapter 5, the DiffeoCFM from Collas, Ju, et al. (2026) is learning to sample points on the tangent space $T_{I_d}\mathcal{P}_d$ at identity, and mapping them back on the manifold. In reality, they transform a training dataset of points on $\mathcal{P}_d$ to velocities from I_d and learn how to sample new velocities. On the contrary, the RiemannianCFM and the proposed WrappedCFM really keep the points on the manifold, and rely on the tangent space to learn the vector field, whose natural space is a tangent space.

Moreover, choosing a single tangent space to project the data often leads to deformation (Yger and Sugiyama 2015; Jaquier, Rozo, and Asfour 2024). On $\mathcal{P}_d$, the Riemannian logarithm is not an isometric mapping and thus, does not preserve distances. This is once again very visible in the synthetic experiment on the different Flow Matching methods in Section 5.5: the DiffeoCFM is struggling to reproduce badly conditioned SPD matrices (see Figure 5.6). This also affects the wrapped Gaussian distribution that includes the Jacobian determinant of the exponential mapping to compensate for this deformation.

8.2.3 When the theory meets the applications

Initially, the wrapped Gaussian was built as a purely theoretical probability distribution in Chapter 3. The goal was to build a geometry-aware anisotropic distribution. Then, we saw that this model had nice theoretical applications: it is the right object to build the probabilistic framework for extrinsic learning tools on $\mathcal{P}_d$, encompassing classifiers or kernels as we saw in Chapter 4, but also Flow Matching methods from Chapter 5. Thus, the wrapped Gaussian is a handy tool when building theoretical frameworks, it helps unify different tools that could *a priori* seem independent.

However, we still wanted our wrapped Gaussian to be useful in practice, as one of the motivations is Brain-Computer Interface (BCI). We quickly saw that estimating the parameters of the full wrapped Gaussian model was difficult in the synthetic experiments of Section 3.3.3. Indeed, having a good estimation error required a tremendous amount of samples, which is difficult to achieve in BCI. We showed that restricting the dispersion matrix Σ to be diagonal, reducing the number of coefficients to estimate from $\mathcal{O}(d^4)$ to $\mathcal{O}(d^2)$ could lead to a lower estimation error.

This estimation problem arose again in Chapter 4, particularly in the numerical comparison of the different classifiers presented in Section 4.3.5. On datasets with a large number of samples, the proposed *Wrapped Discriminant Analysis* algorithms generally outperformed the other methods. However, when the number of samples was small, these algorithms struggled to accurately estimate the parameters of the wrapped Gaussian model, which consequently led to degraded classification performance (see Table 4.3).

Then, in Chapter 6, we tackled a real BCI problem, where the data available was limited, thus, as noted above, the wrapped Gaussian does not seem very suited for this problem. However, we were able to simplify the model by centering the Gaussian in the tangent space by setting $\mu = 0$ and considering a diagonal dispersion matrix Σ. Then, the estimation problem simplifies a lot, as a "good" estimator of the anchor $\boldsymbol{P}$ of the wrapped Gaussian is now the Fréchet mean, and once this is computed, the dispersion matrix Σ can be computed in a Euclidean manner (after projection in the tangent space at the Fréchet mean). Thus, even if the wrapped Gaussian is more of a theoretical tool, this model can still be interesting and lead to good results in practice, once simplified.

8.2.4 What really makes the wrapped Gaussian useful?

As we saw above, and as detailed in Chapter 4 and Chapter 6, assuming enough training data or a simplified model, the wrapped Gaussian is nevertheless capable of being useful in practice. Here, we want to give two main arguments for why the statistical model can be useful.

- The first argument is that, using the wrapped Gaussian, the choice of the tangent space becomes a real parameter of the statistical model, parameterized by the anchor of the wrapped Gaussian $\boldsymbol{P}$. Thus, this parameter can be class specific, as in the Wrapped Discriminant Analysis of Section 4.3.3 where each class relies on its own tangent space or can change over time as in the WrappedCFM of Section 5.4. For these two examples, their respective already studied counterparts (respectively the Tangent Space LDA and the DiffeoCFM) used only one tangent space to map the whole data onto. The problems of using a single tangent space have been described in Section 8.2.2 and is also addressed in Section 7.2 of Jaquier and Rozo (2027) and in Jaquier, Rozo, and Asfour (2024). Learning the best tangent space to map the data has been studied in several works, such as Yger and Sugiyama (2015), Huang, Ruiping Wang, Shan, X. Li, et al. (2015) and Z. Chen, Yue Song, T. Xu, et al. (2024). When using the wrapped Gaussian, one can optimize over the anchor point $\boldsymbol{P}$, discriminate over $\boldsymbol{P}$ and learn or tune this parameter directly in the model. This parameter is fully encompassed in the model, and does not rely on any external procedure.
- The second main argument for the wrapped Gaussian is its anisotropy. This is the main difference with the isotropic Gaussian from Said, Bombrun, et al. (2017), and was one of our main goals when building the wrapped Gaussian (see Section 8.2.1). We strongly believe that real data is not isotropic, thus using an anisotropic distribution will allow us to better model the data. This is what we started to show in Section 6.6.3 for the One-Class classifiers: the gain obtained by the proposed One-Class WG is probably due to its anisotropy.

Once these two arguments are combined, we have what makes the wrapped Gaussian

Dataset	MDM	TS-LDA	Ho-WDA
Indiana	$58.01_{\pm 0.72}$	$67.07_{\pm 0.53}$	$\mathbf{73.74}_{\pm 0.51}$
Pavia Uni.	$72.32_{\pm 0.59}$	$84.61_{\pm 0.06}$	$\mathbf{87.36}_{\pm 0.07}$
Salinas	$36.42_{\pm 0.12}$	$46.30_{\pm 0.17}$	$\mathbf{71.20}_{\pm 0.33}$
Textile	$83.08_{\pm 0.62}$	$83.12_{\pm 0.63}$	$\mathbf{86.26}_{\pm 0.59}$
BreizhCrops	$45.48_{\pm 0.25}$	$47.67_{\pm 0.32}$	$\mathbf{54.66}_{\pm 0.45}$

Table 8.1. Partial reproduction of Table 4.3. Classification accuracy (%, mean ± standard deviation) of (some of) the different classifiers considered in Section 4.3 on (some of) the benchmark datasets. The best result for each dataset is highlighted in bold; cells are shaded red (worst) to green (best) by within-row rank.

reliable and interesting in practice. In order to illustrate the benefits of both arguments, let us look back at the classification results obtained for the numerical experiment comparing the different classifiers (MDM, TS-LDA/QDA, Ho-WDA and He-WDA) in Section 4.3.5. For simplicity, we reproduced part of Table 4.3 in Table 8.1, and we added colors corresponding to the rank of the different accuracies. We can clearly see the following pattern appearing:

$$\text{MDM} < \text{TS-LDA} < \text{Ho-WDA}.$$

This clearly illustrates the two advantages of the wrapped Gaussian: the MDM is an **isotropic** algorithm (see Remark 4.4), the TS-LDA is **anisotropic** but maps all the data to only a **single** tangent space (see Proposition 4.5) while the proposed Ho-WDA is both **anisotropic** and maps the data to **different** tangent spaces according to their class.

8.3 Limitations

We have already addressed some of the limitations of this work in the different chapters. We try to compile them here and give more details on the different limitations.

8.3.1 Complexity of the model

A first major limitation of the wrapped Gaussian model that has already been discussed in several ways is its complexity. Indeed, let us recall that the full wrapped Gaussian statistical model depends on three parameters: $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ and $\Sigma \in \mathcal{P}_D$. Thus, the total number of coefficients is

$$\underbrace{\frac{d(d+1)}{2}}_{\boldsymbol{P}} + \underbrace{\frac{d(d+1)}{2}}_{\mu} + \underbrace{\frac{d^2(d+1)^2 + 2d(d+1)}{8}}_{\Sigma},$$

which grows at a rate of $\mathcal{O}(d^4)$. For example, if $d = 10$, there are $1,650$ coefficients to estimate and if $d = 30$, the number jumps to $109,275$. This high dimension is related to the anisotropy we want in our model: the samples are $d \times d$ SPD matrices, thus have $D = d(d+1)/2$ coefficients, and if one wants to encode the covariances between these coefficients, one needs $D(D+1)/2 = (d^2(d+1)^2 + 2d(d+1))/8$ coefficients.

This is one of the main causes of the difficulty of estimating the coefficients of the wrapped Gaussian as we saw in Section 3.3. Having an acceptable estimation error requires an enormous amount of samples. In our application of BCI, we saw that having such an amount of training samples could be difficult, limiting the usage of the full model of the wrapped Gaussian. We already discussed the theory meeting the applications in Section 8.2.3 and some of the solutions to alleviate this problem.

8.3.2 Numerical complexity

Another important problem of the wrapped Gaussian that was not discussed deeply in the previous chapters is the numerical complexity of the wrapped Gaussian. We recall that in order to sample a point from $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$, one needs to sample a point $t \sim \mathcal{N}(\mu, \Sigma)$ and then to compute $\boldsymbol{X} = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t))$ (see Algorithm 1). We will not elaborate on the difficulty and cost of sampling $t \sim \mathcal{N}(\mu, \Sigma)$ (we reviewed some of the existing algorithms in the introduction of Chapter 5). While computing $\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t)$ is not computationally expensive using Proposition 2.33, we know from Proposition 2.19 that computing the AIRM exponential mapping is expensive. Indeed, it requires at least two eigendecompositions, which are known to be expensive.

Moreover, most of the operations related to wrapped Gaussians rely on computing a Riemannian exponential or logarithm. We did not emphasize on this, but the computational complexity of the WrappedCFM method developed in Chapter 5 to do Riemannian Flow Matching is quite high. Indeed, computing at each time step t the conditional flow of Equation (5.13) and the corresponding conditional vector field given in Theorem 5.6 requires the computation of several matrix logarithms and matrix square roots, which are expensive operations. We can see this effect in the training and sampling times given in Appendix B.2: the proposed WrappedCFM is the slowest model to train and to sample. Compared to the RiemannianCFM, the WrappedCFM is always slower to train, with a gap ranging from 2% to 11%, which is still acceptable.

This numerical complexity is inherent to the choice of the AIRM geometry, and is a known limitation of using this geometry in practice.

8.3.3 Limitations of interpretability tools

In Chapter 7, we built and discussed different interpretability tools to better understand the decisions taken by a Riemannian classifier. These tools are mainly limited as we do not have a proper **ground truth** to compare our findings to. Understanding an interpretation given for an output thus frequently requires expert knowledge. Moreover, as pointed out by Lipton (2018), *post-hoc* explanations, as the ones we propose in Chapter 7, can be misleading. As humans, we tend to rationalize our choices after making them, thus, a model (or a human) can produce a plausible explanation, but that does not reflect the real decision process. Thus, Lipton (2018) "caution[s] against blindly embracing post-hoc notions of interpretability, especially when optimized to placate subjective demands."

Finally, we mainly interpreted in Chapter 7 *backward models* as defined by Haufe et al. (2014) as the model that tries to reverse the data generation process. In his paper Haufe et al. (2014), he draws attention to the fact that "significant nonzero weights may also be observed at channels the activity of which is statistically independent of the brain process under study". This means that our method are also sensitive to noise, and one must make sure that what we are interpreting as "useful information" is not simply noise independent form the brain process being decoded. Future works could make sure that the proposed methods do not suffer from this flaw. If it is the case, Haufe et al. (2014) recommend transforming the weight vector back into an activation pattern before attempting any neurophysiological interpretation. Finding such transformations is not trivial, thus future works could focus on finding how to go from the weights to the activation patterns.

8.4 Perspectives

The writing of this manuscript gave me the opportunity to take a fresh look at problems I had worked on for the past three years. This fresh look sparked new questions and new ideas on my work. Some of these ideas have been developed in this manuscript, such as a more detailed

analysis of the Maximum Likelihood Estimator of the wrapped Gaussian in Section 3.3, a conjecture for the identifiability of the wrapped Gaussian on a general Hadamard space in Section 3.4, a study of different Riemannian kernels in Section 4.5 or the speed-up of the Riemannian t-SNE algorithm as detailed in Section 7.2.2. Other ideas did not make it into this manuscript, as they are not mature enough at the time of writing it.

In this section, I tried to provide different perspectives for the work done in this thesis, organizing them on a virtual timeline, from works that can be done directly to more distant, long-term ideas.

8.4.1 Short-term ideas

In this first section, we will discuss ideas that are directly inspired by the work done in the manuscript.

Estimation of the parameters of the wrapped Gaussian In Section 3.3, we focus on estimating the parameters of a wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ given a finite sample $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$. We tackle this problem by using a *Maximum Likelihood Estimator* (MLE). We saw numerically in Section 3.3.3 that this method[4] did not manage to get a good estimation of the parameters. Let us recall that for the full estimation of the three parameters, we did not even achieve an estimation error below 0.1 with $10,000$ samples, even in dimension $d = 2$. This clearly shows that there is a problem somewhere. A first goal would be to address the issue. For this, we provide here several ideas.

- **An asymptotic analysis of the Jacobian determinant $J_{\boldsymbol{P}}$.** Recall from Proposition 3.32 that, when $\mu = 0_D$ and $\Sigma = I_d$, the MLE $\hat{\boldsymbol{P}}_N$ of $\boldsymbol{P}$ is the solution of the following optimization problem:

$$\hat{\boldsymbol{P}} \in \operatorname*{arg\,min}_{\boldsymbol{P} \in \mathcal{P}_d} \sum_{i=1}^{N} \frac{1}{2} \delta(\boldsymbol{P}, \boldsymbol{X}_i)^2 + \log J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i)). \tag{8.2}$$

  Thus, the MLE of $\boldsymbol{P}$ is the Fréchet mean penalized by the sum of the log of the Jacobian determinants $\sum_{i=1}^{N} \log J_{\boldsymbol{P}}(\mathrm{Log}_{\boldsymbol{P}}(\boldsymbol{X}_i))$. This term also appears for the general MLE of $\boldsymbol{P}$ given in Proposition 3.36. It encodes the deformation induced by the mapping of the $\boldsymbol{X}_i$'s from the manifold $\mathcal{P}_d$ to the tangent space $T_{\boldsymbol{P}}\mathcal{P}_d$ at $\boldsymbol{P}$. Intuitively, especially in the case of an isotropic wrapped Gaussian $\mathrm{WG}(\boldsymbol{P}; 0_D, I_D)$, it seems coherent that the deformation will be minimized at the Fréchet mean, as it corresponds to the mean of the distribution in this case according to Proposition 3.10. Moreover, numerically, we did not observe any important differences between using the Fréchet mean and solving Equation (8.2) when trying to estimate $\boldsymbol{P}$.
  An asymptotic analysis of this log of Jacobian determinants could help us quantify the importance of this term in the optimization problem, and better understand how the MLE of the wrapped Gaussian is related to classical estimators such as the Fréchet mean or the Euclidean MLE of a multivariate Gaussian.
- **Use other estimators.** In this work, we have studied only the classical MLE and methods of moments in order to estimate the parameters of the wrapped Gaussian. However, the literature on statistical estimators is large (Casella and R. Berger 2024), and perhaps more appropriate estimators exist for our problem. In particular, Bayesian statistics (Gelman et al. 1995) could be of great help. The difficulty will then be to choose the best priors for the different parameters in order to use a *maximum a posteriori* (MAP) estimator. We could start by trying to extend classical priors used for the multivariate Gaussian, such as a Normal–inverse-Wishart prior on (μ, Σ) (K. P.

[4] Or at least the implementation of this estimation we used.

Murphy 2007) or an independent Gaussian prior on μ and an inverse-Wishart prior on Σ. These priors suppose some kind of knowledge on the distributions of the parameters. A common noninformative prior used is the Jeffreys prior from Jeffreys (1946), which is proportional to the square root of the determinant of the Fisher information matrix (Kay 1997; S.-i. Amari and Nagaoka 2000). Computing the Fisher information matrix in closed form for the wrapped Gaussian might be ambitious; however, approximating it numerically may be enough.

General results on the identifiability of the wrapped Gaussian In Section 3.4, we have extended the definition of the wrapped Gaussian from the manifold $\mathcal{P}_d$ of SPD matrices to a general Hadamard manifold $\mathcal{M}$. We focus on three Hadamard manifolds: the flat Euclidean space $\mathbb{R}^d$, the manifold of SPD matrices $\mathcal{P}_d$ and the hyperbolic space $\mathbb{H}^d$. We saw that the wrapped Gaussian was **not** identifiable on $\mathbb{R}^d$ and on $\mathcal{P}_d$, while it was identifiable on $\mathbb{H}^d$. Thanks to these results, we proposed a conjecture on the identifiability of the wrapped Gaussian on a general Hadamard manifold in conjecture 3.45. This conjecture is based on the de Rham decomposition of $\mathcal{M}$ given in Theorem 3.43. Proving[5] this conjecture would help close the theoretical study of the wrapped Gaussian. We have several different results on the identifiability of the wrapped Gaussian depending on the space, and having a higher-level theorem linking them all together would give a better understanding of the specific results on each space.

Statistical tests using the wrapped Gaussian In Chapter 6, we consider two One-Class classifiers: the *One-Class MDM* and the new *One-Class WG*. They are both trained on the class `awake`, on which they estimate the parameters of an isotropic Gaussian (in practice, only the center of mass of the isotropic Gaussian is estimated) for the One-Class MDM and the parameters of a wrapped Gaussian for the One-Class WG. Then, when a new unlabeled point $\boldsymbol{Z}$ is given, it is classified based on a threshold on the likelihood under the estimated model. Let us focus on the proposed One-Class WG, but similar work could be done for the One-Class MDM. The problem we are trying to solve here, at classification time, is a statistical question: does $\boldsymbol{Z}$ belong to the distribution $\mathrm{WG}(\hat{\boldsymbol{X}}_N; 0_D, \hat{\Sigma}_N)$? More generally, one could consider the following statistical test:

$$H_0 \colon \boldsymbol{Z} \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) \quad \text{against } H_1 \colon \boldsymbol{Z} \nsim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma).$$

Then, classifying $\boldsymbol{Z}$ as `awake` is accepting H_0; classifying it as `anesthetized` is rejecting it. The next step is to derive a statistic and to deduce from it a rejection region. In Chapter 6, the statistic we chose was the likelihood under the estimated model, and the rejection region was built from a fixed threshold. This construction is quite *ad hoc*, and is inspired by the Riemannian potato from Barachant, Andreev, et al. (2013) studied in Section 4.4.

In the classical Euclidean case, testing whether $t \sim \mathcal{N}(\mu, \Sigma)$, where μ and Σ are unknown and estimated from a finite sample, is the classical outlier test for a multivariate normal population. The statistic used for this test is the *multivariate Hotelling* T^2 from Hotelling (1992), with its law under H_0:

$$F = \frac{N(N-d)}{d(N^2-1)}(t - \bar{X}_N)^\top \hat{\Sigma}_N^{-1}(t - \bar{X}_N) \sim F_{d,N-d} \text{ under } H_0,$$

where $\bar{X}_N$ and $\hat{\Sigma}_N$ are the classical MLEs for the multivariate Gaussian, and $F_{d,N-d}$ is the F-distribution (Snedecor 1934). With this, one is able to derive a rejection region adapted to the chosen level α. This machinery is described in Chapter 5 of T. W. Anderson (2003), and is based on the likelihood ratio criterion. A first good step in this direction would be to study

[5] Or disproving.

a wrapped version of the one-sample t-test, defined as follows: let $\boldsymbol{X}_1, ..., \boldsymbol{X}_N \sim \mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$, where μ and Σ are unknown[6],

$$H_0 \colon \boldsymbol{P} = \boldsymbol{P}_0 \quad \text{against } H_1 \colon \boldsymbol{P} \neq \boldsymbol{P}_0.$$

Classically, using the Neyman–Pearson lemma (Neyman and Pearson 1992), one would consider the likelihood ratio as a statistic for this test. This would require studying first the problem listed above, around the analysis of the MLE of the wrapped Gaussian.

Online Riemannian t-SNE In Section 7.2, we introduced a Riemannian version of the t-SNE algorithm from Maaten and Hinton (2008). As we said in Remark 7.2, the t-SNE is an offline transductive algorithm: one can only compute the low-dimensional embedding of a fixed point cloud, and if a new point is added to the dataset, one needs to recompute the full embedding. In BCI, feedback is a crucial part of a well-functioning system (Lotte, Larrue, et al. 2013). Classical forms of feedback in a motor imagery paradigm are simply a screen that displays a bar or a cursor whose direction depends on the task recognized by the classifier and whose size is proportional to the confidence of the classifier in the recognized tasks (G. Pfurtscheller and C. Neuper 2001).

Using the Riemannian t-SNE, one could imagine a new type of feedback: an online visualization of the brain state, as recorded by the EEG and understood through Riemannian geometry. Such a visualization would take the form of an online t-SNE embedding showing the subject's current mental state, alongside "targets" for the different tasks. This would require an online version of t-SNE. As we began to describe in the conclusion of Chapter 7, and inspired by H. Ma and K. Chen (2024), one approach would be to compute the embedding $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_N$ of a calibration dataset $\boldsymbol{X}_1, ..., \boldsymbol{X}_N$ offline beforehand, and then, once a new high-dimensional point $\tilde{\boldsymbol{X}}$ arrives, optimize only the embedding $\tilde{\boldsymbol{Y}}$ of this new point, keeping the previously computed embedding fixed. This small optimization problem could be solved very quickly[7], while still producing a coherent embedding. The full embedding might need to be re-optimized every 10 to 15 newly added points, in order to preserve a coherent structure.

Once the study of this online Riemannian t-SNE is done, a real-life BCI experiment could be set up, in order to assess the value of this feedback method. This would truly complete the loop of this thesis: from the application to the theory, back to the application. Duan et al. (2021) has already tried a similar experiment, using Diffusion Map (Coifman et al. 2005) to visualize the feedback. In this study, no solid, reliable evidence of learning was found in most of the subjects. As our proposed tool is geometry-aware, thus better reflecting the geometry used by the classifiers, we have hopes that using the Riemannian t-SNE can improve the learning capabilities of subjects.

8.4.2 Long-term ideas

After discussing some direct follow-ups of the work developed in this thesis, we will give some broader ideas that could be studied in the light of this manuscript, or more generally that are inspired by the work done these past three years. A chronological order was attempted here, with no guarantee of success.

Mixture of wrapped Gaussians and clustering A classical machine learning result on the Euclidean multivariate Gaussian, recalled in Section 3.9.6 of Goodfellow et al. (2016), is that "A Gaussian mixture model [(GMM)] is a universal approximator of densities, in the sense that any smooth density can be approximated with any specific nonzero amount of error

[6] Maybe, one should start with $\mu = 0$.
[7] Assuming an efficient implementation.

by a Gaussian mixture model with enough components." This statement can be formalized by the following theorem, inspired from T. T. Nguyen et al. (2020):

Theorem 8.1 (GMMs are universal approximators of densities)**.** *For any probability density $f \in L^1(\mathbb{R}^d)$ and any $\varepsilon > 0$, there exists a finite Gaussian mixture*

$$g = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k),$$

such that $\|f - g\|_1 \leqslant \varepsilon$.

One could imagine a similar result for the wrapped Gaussian. The proof of Theorem 8.1 relies on functional analysis arguments, uses the density of continuous compactly supported functions in $L^1(\mathbb{R}^d)$, a well-chosen mollifier (the Gaussian kernel) and a discretization of the mixing measure. At the time of writing this manuscript, the exact extension to the wrapped Gaussian is not known. More precisely, the Euclidean proof deeply uses convolution, and its extension to $\mathcal{P}_d$ is unclear to the author.

From a more applied point of view, estimating a mixture of Gaussians requires the use of an *Expectation-Maximization* (EM) algorithm as described by Dempster et al. (1977). This iterative algorithm alternates between estimating the weights of each component and the parameters of each Gaussian. An EM algorithm for wrapped Gaussians (with fixed $\mu = 0_D$) has been implemented by the author, and doing clustering on real data would be an interesting, yet natural follow-up of the work already done in this manuscript.

Lower rank dispersion matrix As we saw in Section 8.3 and throughout this thesis, one of the main limitations of the wrapped Gaussian is the dimension of its parameters ($\mathcal{O}(d^4)$). The biggest parameter being the dispersion matrix, that has $^{(d^2(d+1)^2+2d(d+1))}/_8$ coefficients. One of the solutions that we discussed in Section 3.3 and applied in Chapter 6 with the One-Class WG is to restrict ourselves to *diagonal* dispersion matrices, reducing the number of coefficients to only $\mathcal{O}(d^2)$.

Another solution could be to restrict ourselves to lower-rank dispersion matrices. In order to avoid any problems when using the multivariate Gaussian, the dispersion matrix is often required to be SPD. A low-rank matrix is not SPD. However, as done in *Factor Analysis* and its special case *Probabilistic PCA* (Lawley and Maxwell 1962; Tipping and Bishop 1999; Ghojogh et al. 2021), we could impose that the dispersion matrix Σ has the following structure:

$$\Sigma = \boldsymbol{L}\boldsymbol{L}^\top + \Psi,$$

where $\boldsymbol{L} \in \mathbb{R}^{d \times r}$ and $\Psi = \operatorname{diag}(\psi_1, ..., \psi_d)$, with $\psi_i > 0$. This decomposes the dispersion matrix into a low- rank part ($\boldsymbol{L}\boldsymbol{L}^\top$) that is positive semidefinite and an SPD diagonal part Ψ. The sum of both parts is SPD, and it adds some structure to our problem.

Several questions then arise: can this model be more easily estimated than the full one? How does one find the right value of r? Is this model better in practice? Several of these questions have been studied in the Factor Analysis literature (Ghojogh et al. 2021), and seeing if it could be adapted to the wrapped Gaussian case is an interesting question. Maybe a solution will come from studying the geometric structure of positive semidefinite matrices of fixed rank (Bonnabel and Sepulchre 2010; Massart and Absil 2020; Massart, Hendrickx, et al. 2019).

Structured data and Geometric Statistics In this manuscript, we focused on a specific distribution, the wrapped Gaussian, on a specific manifold, $\mathcal{P}_d$, the manifold of SPD matrices. We have already started to explore how this specific distribution could be extended to a larger class of manifolds (see Section 3.4). Another possible extension, in a different direction, would

be to study other distributions defined on manifolds. As we said in Section 2.3.4, wrapping a Euclidean distribution onto a manifold can extend any Euclidean distribution to a distribution on a manifold. However, this is not the only way of defining a distribution on a manifold (see, e.g., the isotropic Gaussian from (Said, Bombrun, et al. 2017), the von Mises–Fisher distribution (R. A. Fisher 1953) and all of its generalizations, or the Wishart distribution from Wishart (1928)). The type of distribution studied depends heavily on the type of data and on the application considered. Investigating other distributions defined on Riemannian manifolds, such as *Wrapped Elliptically Contoured Distributions* defined in Section 3.2.5, could, for example, lead to more robust tools for Riemannian data.

Moreover, depending on the application considered, different manifolds come into play: in robotics, one uses the manifold of SPD matrices but also the group of special orthogonal matrices $\mathrm{SO}(d)$ (Jaquier and Rozo 2027); in biology and chemistry, computing new protein designs relies on SE(3)-invariance, where SE(3) is the special Euclidean group (Jumper et al. 2021; Bose et al. 2024; Woo et al. 2026). Archaeology also motivated the definition and study of *Shape Space* (Kendall 1977; Kendall 1984). These different applications and spaces rely on different structures and invariances, and, as we saw throughout this work, taking them into account is important in order to build powerful tools. The main structure used in this work is the symmetric positive definiteness of matrices, with an affine-invariant metric. This structure can be generalized to other types of structure, with other types of invariance. When the symmetry of a problem is too strong, taking a quotient to simplify the space is a powerful tool. In our work, we also encountered a quotient structure when studying equivalent wrapped Gaussians in Section 3.2.4, but we did not study this structure in depth. Defining probability distributions on quotient manifolds or spaces could lead to interesting discoveries.

The world of structured data and the study of probability distributions can be brought together in two ways. The first one is to continue the work done in this manuscript, that is, to study probability distributions defined on structured spaces such as Riemannian manifolds. The second one is to see the set of probability distributions as a structured set itself and to study it using the tools we developed here. Indeed, if we restrict ourselves to the distributions admitting a density, this set can be written as

$$\left\{ f \in L^1(\mathcal{M}) \mid f \geqslant 0 \text{ and } \int_{\mathcal{M}} f(x)\mathrm{dvol}(x) = 1 \right\}.$$

One could then try to apply to this set the knowledge and the arsenal of tools used for structured data. As we saw in the introduction of this manuscript, probability distributions are used to describe and model any dataset. In the work done during this PhD, we mainly focused on a *parametric* distribution, the wrapped Gaussian; however, to model more complex distributions, one would need to focus on *non-parametric* distributions. We started investigating non-parametric distributions in Chapter 5 when studying Riemannian Flow Matching, and continuing to explore generative models for structured data, and beyond, is an interesting perspective (Peyré 2026; Caucheteux et al. 2026).

Another interesting question is how to compare probability distributions. For example, if we model the covariance matrices of different subjects using probability distributions, comparing these distributions can help us cluster the subjects. Building tools to compare these distributions is thus needed. A famous tool that takes into account the structure of the space of probability distributions is *optimal transport*. On manifolds, optimal transport has mainly been studied from a theoretical point of view (see Part II of Villani 2009). Numerical tools have started to appear, such as *Sliced-Wasserstein* on the circle in Bonet, Berg, et al. (2022) or on more general Hadamard manifolds in Bonet, Drumetz, et al. (2025). Optimal transport is a well-known tool for domain adaptation (Courty et al. 2016), and adapting it to structured data can be an interesting perspective. Tools such as *Neural Optimal Transport*, which use a neural network to learn an optimal transport map, have started to be extended to manifolds (Micheli et al. 2026).

Scaling laws in BCI The last perspective discussed here is oriented toward BCI and how to improve the known models. A major takeaway from *Large Language Models* (LLMs) is that the more one has access to data, the better the trained model will be. This is part of the idea behind the *scaling laws* empirically observed in LLMs (Kaplan et al. 2020; Hoffmann et al. 2022) that dictate the relationship between model performance and three main resources: the model size, the dataset size and the compute power.

In Kaifosh and Reardon (2025), the authors showed that a similar scaling law can also be exhibited in *surface electromyography* (sEMG) decoding. In this work, they use a custom sEMG wristband to detect individual motor units for three applications: continuous 1D cursor control via wrist angle, discrete gesture detection (pinches, taps, swipes) and handwriting text entry. They trained different models to decode these different tasks and observed reliable performance improvement as the number of participants added to the training corpus increased. Their largest training corpus was for the handwriting recognition task, for which they had more than 6,000 participants, with each participant having recorded on average more than 1h of data. This is a proof-of-concept that population-scale sEMG data collection and deep learning can produce a generalizable, calibration-free neuromotor interface.

In the same spirit, we discussed EEG foundation models in Section 2.1.1 that require a lot of data to be trained (W.-B. Jiang et al. 2024; El Ouahidi et al. 2025), a data-scaling logic that echoes the empirical trends observed in both LLMs (Kaplan et al. 2020) and sEMG decoding (Kaifosh and Reardon 2025). Having a large-scale approach to BCI, with some very large-scale datasets, could help alleviate the inter-subject variability issue BCI is currently facing. This would require some *transfer learning* and *domain adaptation* expertise in order to benefit from a very large database.

Compared to text data used by LLMs, or sEMG data used in Kaifosh and Reardon (2025), collecting BCI data requires a lot more work (Lotte, Dreyer, et al. 2025). Thus, having large-scale datasets in BCI is a very challenging task, that requires a controlled laboratory setting, careful electrode placement, and a cooperative, motionless participant, all of which severely limit the throughput of data collection. Scaling BCI datasets to the size seen in LLMs or sEMG decoding would therefore require rethinking the acquisition pipeline itself — for instance through cheaper, more ergonomic dry-electrode devices, or through collaborative, multi-site data-sharing efforts across the BCI community — rather than simply scaling up existing protocols.

8.5 Concluding Remarks

This PhD has brought me to very different places, from BCI to Riemannian geometry, that do not seem related at first sight. However, this is, in some sense, one of the goals of mathematics: discovering abstract patterns, learning how they are related to one another, and sometimes solving real-world problems. This is what we tried to do in this thesis: using the very abstract tools from Riemannian geometry, we wanted to better understand how one could use patterns discovered in SPD matrices to infer physical patterns present in EEGs. The mathematical foundations of the tools developed here can help us trust them and their decisions once deployed on real data.

However, I also experienced the gap between the theory and the application. In theory, everything works well, yet, once faced with real, noisy and few data, things often get more complicated and the nice theory can unfortunately vanish. Sometimes, when dealing with real data, having simplifying assumptions will help (see Chapter 6), while not completely satisfying the mathematician.

Thus, when building theoretical tools, a good practice is to keep the application and its complexity in mind. This is one of the main lesson learned during this PhD, and that I will try to keep in mind during my future career.

Bibliography

Albergo, Michael S and Eric Vanden-Eijnden (2022). "Building normalizing flows with stochastic interpolants". In: *arXiv preprint arXiv:2209.15571* (cit. on pp. 112, 251).

Albergo, Michael S., Nicholas M. Boffi, and Eric Vanden-Eijnden (2023). "Stochastic Interpolants: A Unifying Framework for Flows and Diffusions". In: *arXiv preprint arXiv:2303.08797* (cit. on p. 128).

Almeida, D (2015). "Awake and unable to move: what can perioperative practitioners do to avoid accidental awareness under general anaesthesia?" In: *Journal of Perioperative Practice* 25.12, pp. 257–261 (cit. on pp. 24, 135, 136).

Altaheri, Hamdi, Ghulam Muhammad, and Mansour Alsulaiman (2022). "Physics-informed attention temporal convolutional network for EEG-based motor imagery classification". In: *IEEE transactions on industrial informatics* 19.2, pp. 2249–2258 (cit. on pp. 22, 23).

Amari, Shun-ichi (1978). "Feature spaces which admit and detect invariant signal transformations". In: *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 452–456 (cit. on p. 5).

Amari, Shun-ichi and Hiroshi Nagaoka (2000). *Methods of information geometry*. Vol. 191. American Mathematical Soc. (cit. on p. 184).

Amari, SI (1997). "Information geometry". In: *Contemporary Mathematics* 203.81-96, p. 4 (cit. on p. 44).

Anderson, Charles W and Zlatko Sijercic (1996). "Classification of EEG signals from four subjects during five mental tasks". In: *Solving engineering problems with neural networks: proceedings of the conference on engineering applications in neural networks (EANN'96)*. Turkey, pp. 407–414 (cit. on pp. 22, 23).

Anderson, T. W. (2003). *An Introduction to Multivariate Statistical Analysis*. 3rd ed. Hoboken, NJ: Wiley (cit. on p. 184).

Ang, Kai Keng, Zheng Yang Chin, Haihong Zhang, and Cuntai Guan (2008). "Filter Bank Common Spatial Pattern (FBCSP) in Brain-Computer Interface". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 2390–2397. DOI: 10.1109/IJCNN.2008.4634130 (cit. on p. 21).

Aristimunha, Bruno, Raphael Yokoingawa de Camargo, Sylvain Chevallier, Oeslle Lucena, Adam Thomas, M. Jorge Cardoso, Walter Lopez Pinaya, and Jessica Dafflon (2023). "Synthetic Sleep EEG Signal Generation using Latent Diffusion Models". In: *Deep Generative Models for Health Workshop NeurIPS 2023* (cit. on pp. 27, 128).

Aristimunha, Bruno, Igor Carrara, Pierre Guetschel, Sara Sedlar, Pedro Rodrigues, Jan Sosulski, Divyesh Narayanan, Erik Bjareholt, Quentin Barthelemy, Robin Tibor Schirrmeister, Reinmar Kobler, Emmanuel Kalunga, Ludovic Darmet, Cattan Gregoire, Ali Abdul Hussain, Ramiro Gatti, Vladislav Goncharenko, Anton Andreev, Alberto Tates, Simon Kojima, Jordy Thielen, Davoud Hajhassani, Katelyn Begany, Thomas Moreau, Yannick Roy, Vinay Jayaram, Alexandre Barachant, and Sylvain Chevallier (2026). *Mother of all BCI Benchmarks*. Version 1.5.0. DOI: 10.5281/zenodo.10034223 (cit. on pp. 12, 22, 23, 26, 30, 101, 108, 238).

Aristimunha, Bruno, Pierre Guetschel, Martin Wimpff, Lukas Gemein, Cedric Rommel, Hubert Banville, Maciej Sliwowski, Daniel Wilson, Simon Brandt, Théo Gnassounou, Joseph Paillard, Bruna Junqueira Lopes, Sara Sedlar, Thomas Moreau, Sylvain Chevallier, Alexandre Gramfort, and Robin Tibor Schirrmeister (2026). *Braindecode: toolbox for decoding raw electrophysiological brain data with deep learning models*. Zenodo. DOI: 10.5281/zenodo.17699192 (cit. on pp. 22, 23).

Aristimunha, Bruno, Ce Ju, Antoine Collas, Florent Bouchard, Ammar Mian, Bertrand Thirion, Sylvain Chevallier, and Reinmar Kobler (2026). "SPDlearn: A Geometric Deep Learning Python Library for Neural Decoding Through Trivialization". In: *To be submitted* (cit. on p. 170).

Aristimunha Pinto, Bruno (Mar. 2026). "Learning Representations of Electroencephalogram using Deep Learning". Theses. Université Paris-Saclay ; Universidade Federal do ABC (cit. on p. 22).

Arsigny, Vincent, Pierre Fillard, Xavier Pennec, and Nicholas Ayache (Jan. 1, 2006a). "Geometric Means in a Novel Vector Space Structure on Symmetric Positive-Definite Matrices". In: *SIAM J. Matrix Analysis Applications* 29, pp. 328–347. DOI: 10.1137/050637996 (cit. on pp. 4, 33, 38, 39, 48, 74, 245, 248).

— (2006b). "Log-Euclidean Metrics for Fast and Simple Calculus on Diffusion Tensors". In: *Magnetic Resonance in Medicine* 56.2, pp. 411–421. DOI: 10.1002/mrm.20965 (cit. on pp. 34, 38, 39, 89, 245).

Assran, Mahmoud, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas (2023). "Self-supervised learning from images with a joint-embedding predictive architecture". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629 (cit. on p. 1).

Augenstein, Luis, Noémie Jaquier, Tamim Asfour, and Leonel Rozo (2024). "On Probabilistic Pullback Metrics for Latent Hyperbolic Manifolds". In: *arXiv preprint arXiv:2410.20850* (cit. on p. 89).

Avilov, Oleksii, Sébastien Rimbert, Anton Popov, and Laurent Bougrain (2021). "Optimizing motor intention detection with deep learning: towards management of intraoperative awareness". In: *IEEE Transactions on Biomedical Engineering* 68.10, pp. 3087–3097 (cit. on p. 136).

Ayadi, Imen, Florent Bouchard, and Frédéric Pascal (Sept. 2023). "T-WDA: A Novel Discriminant Analysis Applied to EEG Classification". In: *EUSIPCO*. Helsinki, Finland (cit. on pp. 53, 101).

— (Nov. 2024). *Elliptical Wishart Distributions: Information Geometry, Maximum Likelihood Estimator, Performance Analysis and Statistical Learning* (cit. on pp. 53, 101).

Bach, F. (2024). *Learning Theory from First Principles*. Adaptive Computation and Machine Learning series. MIT Press (cit. on p. 46).

Banerjee, Arindam, Inderjit S Dhillon, Joydeep Ghosh, Suvrit Sra, and Greg Ridgeway (2005). "Clustering on the Unit Hypersphere using von Mises-Fisher Distributions." In: *Journal of Machine Learning Research* 6.9 (cit. on p. 52).

Barachant, Alexandre (Mar. 2012). "Commande robuste d'un effecteur par une interface cerveau machine EEG asynchrone". Theses. Université de Grenoble (cit. on p. 46).

Barachant, Alexandre, Anton Andreev, and Marco Congedo (Jan. 2013). "The Riemannian Potato: an automatic and adaptive artifact detection method for online experiments using Riemannian geometry". In: *TOBI Workshop lV*. Sion, Switzerland, pp. 19–20 (cit. on pp. 104, 135, 137, 176, 177, 184, 251, 255).

Barachant, Alexandre, Quentin Barthélemy, Jean-Rémi King, Alexandre Gramfort, Sylvain Chevallier, Pedro L. C. Rodrigues, Emanuele Olivetti, Vladislav Goncharenko, Gabriel Wagner vom Berg, Ghiles Reguig, Arthur Lebeurrier, Erik Bjäreholt, Maria Sayu Yamamoto, Pierre Clisson, Marie-Constance Corsi, Igor Carrara, Apolline Mellot, Bruna

Junqueira Lopes, Brent Gaisford, Ammar Mian, Anton Andreev, Gregoire Cattan, and Bruno Aristimunha (Jan. 2026). *pyRiemann*. DOI: 10.5281/zenodo.593816 (cit. on pp. 12, 13, 41, 103, 154, 171).

Barachant, Alexandre, Stéphane Bonnet, Marco Congedo, and Christian Jutten (Oct. 2010a). "Common Spatial Pattern revisited by Riemannian Geometry". In: *2010 IEEE International Workshop on Multimedia Signal Processing (MMSP)*. Saint-Malo, France, p. 472. DOI: 10.1109/MMSP.2010.5662067 (cit. on p. 21).

— (Sept. 2010b). "Riemannian geometry applied to BCI classification". In: *Latent Variable Analysis and Signal Separation. 9th International Conference, LVA/ICA 2010, St. Malo, France, September 27-30, 2010, Proceedings*. Ed. by Vigneron, V.; Zarzoso, V.; Moreau, E.; Gribonval, R.; Vincent, and E. (Eds.) Vol. 6365. Series: Lecture Notes in Computer Science. Subseries: Theoretical Computer Science and General Issues. ISBN 978-3-642-15994-7, Softcover. Saint-Malo, France: Springer, pp. 629–636. DOI: 10.1007/978-3-642-15995-4_78 (cit. on pp. 22, 23, 29, 34, 46, 47, 58, 59, 97, 134, 244, 250).

— (2011). "Multiclass brain–computer interface classification by Riemannian geometry". In: *IEEE Transactions on Biomedical Engineering* 59.4, pp. 920–928 (cit. on pp. 10, 23, 46, 47, 58, 59, 97, 98, 137, 164, 176, 250, 260).

— (July 2013). "Classification of covariance matrices using a Riemannian-based kernel for BCI applications". In: *Neurocomputing* 112, pp. 172–178. DOI: 10.1016/j.neucom.2012.12.039 (cit. on pp. 23, 48, 58).

Barbaresco, Frédéric (2019). "Lie group machine learning and Gibbs density on Poincaré unit disk from Souriau Lie groups thermodynamics and SU (1, 1) coadjoint orbits". In: *International Conference on Geometric Science of Information*. Springer, pp. 157–170 (cit. on pp. 53, 55).

Barndorff-Nielsen, O. (1977). "Exponentially Decreasing Distributions for the Logarithm of Particle Size". In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 353.1674, pp. 401–419 (cit. on pp. 53, 58).

Barnes, Josh and Piet Hut (1986). "A hierarchical O (N log N) force-calculation algorithm". In: *nature* 324.6096, pp. 446–449 (cit. on p. 157).

Barnes, Lachlan D, Kevin Lee, Andreas W Kempa-Liehr, and Luke E Hallum (2022). "Detection of sleep apnea from single-channel electroencephalogram (EEG) using an explainable convolutional neural network (CNN)". In: *PLOS one* 17.9, e0272167 (cit. on p. 134).

Barthélemy, Quentin, Louis Mayaud, David Ojeda, and Marco Congedo (2019). "The Riemannian potato field: a tool for online signal quality index of EEG". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 27.2, pp. 244–255 (cit. on pp. 105, 135, 176).

Başar, Erol, C. Başar-Eroglu, S. Karakaş, and M. Schürmann (2001). "Brain oscillations in perception and memory". In: *International Journal of Psychophysiology* 39.2–3, pp. 167–195 (cit. on p. 21).

Battiti, Roberto (1994). "Using mutual information for selecting features in supervised neural net learning". In: *IEEE Transactions on neural networks* 5.4, pp. 537–550 (cit. on p. 21).

Behzad, Ramina and Aida Behzad (2021). "The role of EEG in the diagnosis and management of patients with sleep disorders". In: *J. Behav. Brain Sci* 11.10, pp. 257–266 (cit. on p. 24).

Belkin, Mikhail and Partha Niyogi (2001). "Laplacian eigenmaps and spectral techniques for embedding and clustering". In: *Advances in neural information processing systems* 14 (cit. on p. 49).

Belkina, Anna C, Christopher O Ciccolella, Rina Anno, Richard Halpert, Josef Spidlen, and Jennifer E Snyder-Cappione (2019). "Automated optimized parameters for T-distributed stochastic neighbor embedding improve visualization and analysis of large datasets". In: *Nature communications* 10.1, p. 5415 (cit. on pp. 157, 259).

Benaroch, Camille, Khadijeh Sadatnejad, Aline Roc, Aurélien Appriou, Thibaut Monseigne, Smeety Pramij, Jelena Mladenovic, Léa Pillette, Camille Jeunet, and Fabien Lotte (2021). "Long-Term BCI Training of a Tetraplegic User: Adaptive Riemannian Classifiers and User Training". In: *Frontiers in Human Neuroscience* 15. DOI: 10.3389/fnhum.2021.635653 (cit. on p. 26).

Bendokat, Thomas, Ralf Zimmermann, and P-A Absil (2020). "A Grassmann manifold handbook: Basic geometry and computational aspects". In: *arXiv preprint arXiv:2011.13699* (cit. on pp. 49, 53).

Benedetti, Riccardo and Carlo Petronio (1992). *Lectures on hyperbolic geometry*. Springer Science & Business Media (cit. on pp. 58, 88, 89).

Berger, Hans (1929). "Über das elektroenkephalogramm des menschen". In: *Archiv für psychiatrie und nervenkrankheiten* 87.1, pp. 527–570 (cit. on p. 17).

Berger, Marcel (2003). *A panoramic view of Riemannian geometry*. Berlin, Heidelberg: Springer (cit. on p. 2).

Bergmann, Paul, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger (Apr. 2021). "The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection". In: *Int. J. Comput. Vision* 129.4, pp. 1038–1059. DOI: 10.1007/s11263-020-01400-4 (cit. on pp. 102, 108, 125, 238).

Bergsson, Andri and Søren Hauberg (2022). "Visualizing Riemannian data with rie-sne". In: *arXiv preprint arXiv:2203.09253* (cit. on p. 49).

Berndt, Donald J and James Clifford (1994). "Using dynamic time warping to find patterns in time series". In: *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pp. 359–370 (cit. on p. 10).

Bertrand, Olivier, François Perrin, and Jacques Pernier (1985). "A theoretical justification of the average reference in topographic evoked potential studies". In: *Electroencephalography and Clinical Neurophysiology/Evoked Potentials Section* 62.6, pp. 462–464 (cit. on p. 19).

Bhatia, Rajendra (2007). *Positive Definite Matrices*. Princeton Series in Applied Mathematics. Princeton, N.J: Princeton University Press (cit. on pp. 31, 34, 228).

Bhatia, Rajendra, Tanvi Jain, and Yongdo Lim (2019). "On the Bures–Wasserstein distance between positive definite matrices". In: *Expositiones mathematicae* 37.2, pp. 165–191 (cit. on pp. 34, 41, 245).

Bhattacharyya, Anil (1943). "On a measure of divergence between two statistical populations defined by their probability distribution". In: *Bulletin of the Calcutta Mathematical Society* 35, pp. 99–110 (cit. on p. 44).

Biran, Or and Courtenay Cotton (2017). "Explanation and justification in machine learning: A survey". In: *IJCAI-17 workshop on explainable AI (XAI)*. Vol. 8. 1, pp. 8–13 (cit. on p. 147).

Blankertz, Benjamin, Gabriel Curio, and Klaus-Robert Müller (2001). "Classifying single trial EEG: Towards Brain-Computer interfacing". In: *Advances in neural information processing systems* 14 (cit. on pp. 22, 23).

Blankertz, Benjamin, Guido Dornhege, Matthias Krauledat, Michael Schröder, John Williamson, Roderick Murray-Smith, and Klaus-Robert Müller (2010). "The Berlin Brain-Computer Interface presents the novel mental typewriter Hex-o-Spell." In: *Proceedings of the 3rd International Brain-Computer Interface Workshop and Training Course 2006*, pp. 108–109 (cit. on p. 23).

Blankertz, Benjamin, Steven Lemm, Matthias Treder, Stefan Haufe, and Klaus-Robert Müller (2011). "Single-trial analysis and classification of ERP components—a tutorial". In: *NeuroImage* 56.2, pp. 814–825 (cit. on p. 22).

Blankertz, Benjamin, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and Klaus-Robert Muller (2008). "Optimizing spatial filters for robust EEG single-trial analysis". In: *IEEE Signal processing magazine* 25.1, pp. 41–56 (cit. on pp. 21, 150).

Blokland, Yvonne, Jason Farquhar, Jos Lerou, Jo Mourisse, Gert Jan Scheffer, Geert-Jan van Geffen, Loukianos Spyrou, and Jörgen Bruhn (2016). "Decoding motor responses from the EEG during altered states of consciousness induced by propofol". In: *Journal of Neural Engineering* 13.2, p. 026014 (cit. on pp. 134, 136).

Bogachev, Vladimir I. (2007). *Measure Theory*. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-540-34514-5 (cit. on pp. 54, 66, 68, 247).

Bomatter, Philipp, Joseph Paillard, Pilar Garces, Jörg Hipp, and Denis-Alexander Engemann (2024). "Machine learning of brain-specific biomarkers from EEG". In: *EBioMedicine* 106 (cit. on p. 19).

Bonet, Clément, Paul Berg, Nicolas Courty, François Septier, Lucas Drumetz, and Minh-Tan Pham (2022). "Spherical sliced-wasserstein". In: *arXiv preprint arXiv:2206.08780* (cit. on p. 187).

Bonet, Clément, Lucas Drumetz, and Nicolas Courty (2025). "Sliced-Wasserstein distances and flows on Cartan-Hadamard manifolds". In: *Journal of Machine Learning Research* 26.32, pp. 1–76 (cit. on p. 187).

Bonnabel, Silvere and Rodolphe Sepulchre (2010). "Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank". In: *SIAM Journal on Matrix Analysis and Applications* 31.3, pp. 1055–1070 (cit. on p. 186).

Bose, Joey, Tara Akhound-Sadegh, Guillaume Huguet, Kilian FATRAS, Jarrid Rector-Brooks, Cheng-Hao Liu, Andrei Cristian Nica, Maksym Korablyov, Michael M. Bronstein, and Alexander Tong (2024). "SE(3)-Stochastic Flow Matching for Protein Backbone Generation". In: *The Twelfth International Conference on Learning Representations* (cit. on pp. 113, 187).

Bostanov, Vladimir (2004). "BCI competition 2003-data sets Ib and IIb: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram". In: *IEEE Transactions on Biomedical engineering* 51.6, pp. 1057–1061 (cit. on p. 22).

Bouchard, Florent, Ammar Mian, Malik Tiomoko, Guillaume Ginolhac, and Frédéric Pascal (2024). "Random matrix theory improved Fréchet mean of symmetric positive definite matrices". In: *arXiv preprint arXiv:2405.06558* (cit. on p. 102).

Boucherie*, Charlotte, **Thibault de Surrel***, and Florian Yger (2026). "SPDNet-AE: a Compact SPD Representation through Riemannian Autoencoding". In: *ESANN* (cit. on p. 10).

Boumal, Nicolas (Mar. 2023). *An Introduction to Optimization on Smooth Manifolds*. 1st ed. Cambridge University Press. DOI: 10.1017/9781009166164 (cit. on pp. 34, 36, 45, 85, 103, 153, 154, 227, 249, 258).

Box, George EP and Norman R Draper (1987). *Empirical model-building and response surfaces*. John Wiley & Sons (cit. on p. 133).

Box, George EP and Mervin E Muller (1958). "A note on the generation of random normal deviates". In: *The annals of mathematical statistics* 29.2, pp. 610–611 (cit. on pp. 112, 251).

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13 (cit. on p. 85).

Braun, Max, Noémie Jaquier, Leonel Rozo, and Tamim Asfour (2024). "Riemannian flow matching policy for robot motion learning". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5144–5151 (cit. on p. 113).

Bregman, Lev M (1967). "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming". In: *USSR computational mathematics and mathematical physics* 7.3, pp. 200–217 (cit. on p. 44).

Brehmer, Johann, Sönke Behrends, Pim De Haan, and Taco Cohen (2025). "Does equivariance matter at scale?" In: *Transactions on Machine Learning Research* (cit. on p. 5).

Brehmer, Johann, Joey Bose, Pim De Haan, and Taco S Cohen (2023). "Edgi: Equivariant diffusion for planning with embodied agents". In: *Advances in Neural Information Processing Systems* 36, pp. 63818–63834 (cit. on p. 5).

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32 (cit. on pp. 164, 166, 178, 260).

Briët, Jop and Peter Harremoës (2009). "Properties of classical and quantum Jensen-Shannon divergence". In: *Physical Review A—Atomic, Molecular, and Optical Physics* 79.5, p. 052311 (cit. on p. 44).

Brodu, Nicolas, Fabien Lotte, and Anatole Lécuyer (2011). "Comparative study of band-power extraction techniques for motor imagery classification". In: *2011 IEEE symposium on computational intelligence, cognitive algorithms, mind, and brain (CCMB)*. IEEE, pp. 1–6 (cit. on p. 21).

Bronstein, Michael M, Joan Bruna, Taco Cohen, and Petar Veličković (2021). "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges". In: *arXiv preprint arXiv:2104.13478* (cit. on pp. 3, 5).

Bronstein, Michael M, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst (2017). "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4, pp. 18–42 (cit. on p. 5).

Brooks, Daniel A, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord (Dec. 2019). "Riemannian batch normalization for SPD neural networks". In: *Thirty-third Annual Conference on Neural Information Processing Systems*. Vancouver, Canada (cit. on pp. 57, 58).

Bruhn, J, Paul S Myles, R Sneyd, and Michel MRF Struys (2006). "Depth of anaesthesia monitoring: what's available, what's validated and what's next?" In: *BJA: British Journal of Anaesthesia* 97.1, pp. 85–94 (cit. on p. 136).

Bruna, Joan and Stéphane Mallat (2013). "Invariant scattering convolution networks". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1872–1886 (cit. on p. 5).

Bures, Donald (1969). "An extension of Kakutani's theorem on infinite product measures to the tensor product of semifinite w*-algebras". In: *Transactions of the American Mathematical Society* 135, pp. 199–212 (cit. on p. 41).

Busemann, H. (1955). *The Geometry of Geodesics*. Pure and applied mathematics vol. 6. Academic Press (cit. on p. 49).

Cai, Hanqin, Yuchen Lou, Daniel Mckenzie, and Wotao Yin (2021). "A Zeroth-Order Block Coordinate Descent Algorithm for Huge-Scale Black-Box Optimization". In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139, pp. 1193–1203 (cit. on p. 171).

Caldero, Philippe and Jérôme Germoni (2016). *Histoires hédonistes de groupes et de géométries*. Calvage et Mounet (cit. on p. 31).

Carlini, Nicholas, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin (2019). "On Evaluating Adversarial Robustness". In: *arXiv preprint arXiv:1902.06705* (cit. on p. 170).

Carrara, Igor (2024). "Méthodes avancées de traitement des BCI-EEG pour améliorer la performance et la reproductibilité de la classification". PhD thesis. Université Côte d'Azur (cit. on p. 23).

Casella, George and Roger Berger (2024). *Statistical inference*. Chapman and Hall/CRC (cit. on pp. 67, 183).

Caucheteux, Paul, Clément Bonet, and Anna Korba (2026). "A Unifying View of Variational Generative Wasserstein Flows". In: *Forty-third International Conference on Machine Learning* (cit. on p. 187).

Cavarretta, Bastien, **Thibault de Surrel**, Clément W. Royer, and Florian Yger (2026). "Riemannian Black-box Adversarial Attacks on SPD Networks". Under review at *AAAI 2027*. (cit. on pp. 11, 171).

Cavarretta, Bastien, Florentin Goyens, Clément W. Royer, and Florian Yger (2026). *Complexity guarantees and polling strategies for Riemannian direct-search methods* (cit. on p. 171).

Chakraborty, Anirban, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay (Mar. 2021). "A survey on adversarial attacks and defences". In: *CAAI Transactions on Intelligence Technology* 6.1, pp. 25–45. DOI: 10.1049/cit2.12028 (cit. on p. 170).

Chakraborty, Rudrasis, Jose Bouza, Jonathan H. Manton, and Baba C. Vemuri (2022). "ManifoldNet: A Deep Neural Network for Manifold-Valued Data With Applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.2, pp. 799–810. DOI: 10.1109/TPAMI.2020.3003846 (cit. on p. 57).

Chambon, Stanislas, Mathieu N Galtier, Pierrick J Arnal, Gilles Wainrib, and Alexandre Gramfort (2018). "A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26.4, pp. 758–769 (cit. on pp. 22, 23).

Chami, Ines, Albert Gu, Dat P Nguyen, and Christopher Ré (2021). "Horopca: Hyperbolic dimensionality reduction via horospherical projections". In: *International Conference on Machine Learning*. PMLR, pp. 1419–1429 (cit. on p. 49).

Chami, Ines, Zhitao Ying, Christopher Ré, and Jure Leskovec (2019). "Hyperbolic graph convolutional neural networks". In: *Advances in neural information processing systems* 32 (cit. on p. 58).

Chan, David M, Roshan Rao, Forrest Huang, and John F Canny (2018). "t-SNE-CUDA: GPU-accelerated t-SNE and its applications to modern data". In: *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, pp. 330–338 (cit. on pp. 157, 158).

Chandola, Varun, Arindam Banerjee, and Vipin Kumar (July 2009). "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3. DOI: 10.1145/1541880.1541882 (cit. on p. 104).

Chen, Ricky T. Q. and Yaron Lipman (Feb. 26, 2024). *Flow Matching on General Geometries*. DOI: 10.48550/arXiv.2302.03660. URL: http://arxiv.org/abs/2302.03660 (visited on 04/04/2025). Pre-published (cit. on pp. 110, 113, 116, 119, 122, 126, 252).

Chen, Yilun, Ami Wiesel, Yonina C. Eldar, and Alfred O. Hero (2010). "Shrinkage Algorithms for MMSE Covariance Estimation". In: *IEEE Transactions on Signal Processing* 58.10, pp. 5016–5029. DOI: 10.1109/TSP.2010.2053029 (cit. on pp. 26, 29, 102, 238).

Chen, Ziheng, Yue Song, Gaowen Liu, Ramana Rao Kompella, Xiao-Jun Wu, and Nicu Sebe (2024). "Riemannian multinomial logistics regression for SPD neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17086–17096 (cit. on pp. 8, 47, 177).

Chen, Ziheng, Yue Song, Yunmei Liu, and Nicu Sebe (2024). "A Lie Group Approach to Riemannian Batch Normalization". In: *The Twelfth International Conference on Learning Representations* (cit. on p. 57).

Chen, Ziheng, Yue Song, Tianyang Xu, Zhiwu Huang, Xiao-Jun Wu, and Nicu Sebe (2024). "Adaptive Log-Euclidean Metrics for SPD Matrix Learning". In: *IEEE Transactions on Image Processing* 33, pp. 5194–5205. DOI: 10.1109/TIP.2024.3451930 (cit. on pp. 89, 180).

Chevallier, Emmanuel and Nicolas Guigui (Apr. 2020). "A Bi-Invariant Statistical Model Parametrized by Mean and Covariance on Rigid Motions". In: *Entropy* 22.4, p. 432. DOI: 10.3390/e22040432 (cit. on pp. 55, 85).

Chevallier, Emmanuel, Didong Li, Yulong Lu, and David Dunson (Dec. 2022). "Exponential-Wrapped Distributions on Symmetric Spaces". In: *SIAM Journal on Mathematics of Data Science* 4.4, pp. 1347–1368. DOI: 10.1137/21M1461551 (cit. on pp. 55, 58, 69, 85, 88).

Chevallier, Emmanuel, Gabriel Trindade, and Frank Nielsen (Oct. 2025). "Equivariant log-extrinsic statistics on symmetric cones". working paper or preprint (cit. on p. 80).

Chevallier, Sylvain, Emmanuel K Kalunga, Quentin Barthélemy, and Eric Monacelli (2021). "Review of Riemannian distances and divergences, applied to SSVEP-based BCI". In: *Neuroinformatics* 19.1, pp. 93–106 (cit. on pp. 34, 44).

Chi, Yu Mike, Tzyy-Ping Jung, and Gert Cauwenberghs (2010). "Dry-contact and noncontact biopotential electrodes: Methodological review". In: *IEEE reviews in biomedical engineering* 3, pp. 106–119 (cit. on p. 24).

Chikuse, Y. (2012). *Statistics on Special Manifolds*. Lecture Notes in Statistics. Springer New York (cit. on p. 53).

Cho, Seunghyuk, Juyong Lee, Jaesik Park, and Dongwoo Kim (2022). "A Rotated Hyperbolic Wrapped Normal Distribution for Hierarchical Representation Learning". In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (cit. on pp. 54, 58).

Chowdhury, Rasheda Arman, Younes Zerouali, Tanguy Hedrich, Marcel Heers, Eliane Kobayashi, Jean-Marc Lina, and Christophe Grova (2015). "MEG–EEG information fusion and electromagnetic source imaging: from theory to clinical application in epilepsy". In: *Brain topography* 28.6, pp. 785–812 (cit. on p. 24).

Cimarosto, Pietro, Sebastien Velut, Kalou Castillos-Cabrera, Tresols Juan Torre, Raphaëlle N Roy, and Frederic Dehais (2026). "Near-invisible c-VEP-based passive BCI for mental workload monitoring". In: *Journal of Neural Engineering* 23.2, p. 026014 (cit. on p. 24).

Clerc, M., L. Bougrain, and Fabien Lotte (2016). *Brain-Computer Interfaces 1: Methods and Perspectives*. Brain-computer Interfaces vol. 1. Wiley (cit. on pp. 3, 17, 20, 21, 243).

Coelho Rodrigues, Pedro Luiz, Florent Bouchard, Marco Congedo, and Christian Jutten (Sept. 2017). "Dimensionality Reduction for BCI Classification Using Riemannian Geometry". In: *BCI 2017 - 7th International Brain-Computer Interface Conference*. Graz, Austria: Gernot R. Müller-Putz. DOI: 10.3217/978-3-85125-533-1-16 (cit. on p. 50).

Cohen, Taco and Max Welling (2016). "Group equivariant convolutional networks". In: *International conference on machine learning*. PMLR, pp. 2990–2999 (cit. on p. 5).

Coifman, Ronald R, Stephane Lafon, Ann B Lee, Mauro Maggioni, Boaz Nadler, Frederick Warner, and Steven W Zucker (2005). "Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps". In: *Proceedings of the national academy of sciences* 102.21, pp. 7426–7431 (cit. on p. 185).

Collas, Antoine, Florent Bouchard, Arnaud Breloy, Guillaume Ginolhac, Chengfang Ren, and Jean-Philippe Ovarlez (2021). "Probabilistic PCA From Heteroscedastic Signals: Geometric Framework and Application to Clustering". In: *IEEE Transactions on Signal Processing* 69, pp. 6546–6560. DOI: 10.1109/TSP.2021.3130997 (cit. on p. 102).

Collas, Antoine, Ce Ju, Nicolas Salvy, and Bertrand Thirion (2026). "Riemannian flow matching for brain connectivity matrices via pullback geometry". In: *Advances in neural information processing systems* 38, pp. 59589–59639 (cit. on pp. 8, 89, 113, 116, 117, 125, 177, 179, 238, 252, 254).

Collett, D. and T. Lewis (1981). "Discriminating Between the Von Mises and Wrapped Normal Distributions". In: *Australian Journal of Statistics* 23.1, pp. 73–79. DOI: 10.1111/j.1467-842X.1981.tb00763.x (cit. on pp. 54, 58).

Congedo, Marco, Alexandre Barachant, and Rajendra Bhatia (2017). "Riemannian geometry for EEG-based brain-computer interfaces; a primer and a review". In: *Brain-Computer Interfaces* 4.3, pp. 155–174 (cit. on pp. 4, 30, 244).

Courty, Nicolas, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy (2016). "Optimal transport for domain adaptation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.9, pp. 1853–1865 (cit. on p. 187).

Cover, Thomas and Peter Hart (1967). "Nearest neighbor pattern classification". In: *IEEE transactions on information theory* 13.1, pp. 21–27 (cit. on p. 1).

Criscitiello, Christopher and Nicolas Boumal (2023). "An accelerated first-order method for non-convex optimization on manifolds". In: *Foundations of Computational Mathematics* 23.4, pp. 1433–1509 (cit. on p. 37).

Cueva*, Valérie Marissens, **Thibault de Surrel***, Laurent Bougrain, Seyed Javad Bidgoli, Guy M Cheron, Ana Maria Cebolla Alvarez, Claude Meistelman, Fabien Lotte, Florian Yger, and Sébastien Rimbert (2026). "Wrapped One-Class Riemannian EEG Classifier for BCI-Based Detection of Anesthetic States". In: *10th International Graz Brain-Computer Interface Conference* (cit. on pp. 9, 135).

Cuturi, Marco (2009). "Positive definite kernels in machine learning". In: *arXiv preprint arXiv:0911.5367* (cit. on p. 106).

Daletskii, Ju L and Selim Grigorievich Krein (1965). "Integration and differentiation of functions of Hermitian operators and applications to the theory of perturbations". In: *AMS Translations (2)* 47.1-30, pp. 10–1090 (cit. on p. 228).

Darley, Gatien and Stéphane Bonnet (Aug. 2025). "CNN-SPDNet : un réseau de neurones Riemannien profond pour l'interprétation spatio-fréquentielle en EEG BCI". In: *Gretsi'25 - 30ème Colloque Francophone de Traitement du Signal et des Images*. Strasbourg, France, p. ID1469 (cit. on p. 57).

Davis, Philip J. (1959). "Leonhard Euler's Integral: A Historical Profile of the Gamma Function: In Memoriam: Milton Abramowitz". In: *The American Mathematical Monthly* 66.10, pp. 849–869 (cit. on pp. 73, 74).

Davoudi, Alireza, Saeed Shiry Ghidary, and Khadijeh Sadatnejad (Mar. 2017). "Dimensionality Reduction Based on Distance Preservation to Local Mean for Symmetric Positive Definite Matrices and Its Application in Brain–Computer Interfaces". In: *Journal of Neural Engineering* 14.3, p. 036019. DOI: 10.1088/1741-2552/aa61bb (cit. on p. 50).

de Rham, George (1952). "Sur la réductibilité d'un espace de Riemann." In: *Commentarii mathematici Helvetici* 26, pp. 328–344 (cit. on p. 89).

de Surrel, **Thibault** (2025). "Learning context invariant representations for EEG data". In: *Science Talks* 13, p. 100422. DOI: https://doi.org/10.1016/j.sctalk.2025.100422 (cit. on p. 11).

de Surrel, **Thibault**, Sylvain Chevallier, Fabien Lotte, and Florian Yger (2025). "Geometry-Aware visualization of high dimensional Symmetric Positive Definite matrices". In: *Transactions on Machine Learning Research* (cit. on pp. 10, 151).

— (2026). "Wrapped Gaussian on Hadamard Manifolds". Under review at *Foundations of Computational Mathematics* (cit. on pp. 8, 65).

de Surrel, **Thibault**, Fabien Lotte, Sylvain Chevallier, and Florian Yger (2025). "Wrapped Gaussian on the manifold of Symmetric Positive Definite Matrices". In: *Forty-second International Conference on Machine Learning* (cit. on pp. 7, 8, 65, 81, 95).

de Surrel, **Thibault**, Tristan Venot, Marie-Constance Corsi, and Florian Yger (2025). "Interpretability of Riemannian tools used in Brain Computer Interfaces". In: *2025 IEEE 35th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, pp. 1–6 (cit. on pp. 10, 151).

de Surrel, **Thibault**, Florian Yger, Sylvain Chevallier, and Fabien Lotte (2024). "Averaging trajectories on the manifold of symmetric positive definite matrices". In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1182–1186 (cit. on p. 10).

de Surrel, **Thibault**, Florian Yger, Fabien Lotte, and Sylvain Chevallier (2025). "A probabilistic view on Riemannian machine learning models for SPD matrices". In: *International Conference on Geometric Science of Information*. Springer, pp. 142–151 (cit. on pp. 8, 95).

Debener, Stefan, Falk Minow, Reiner Emkes, Katharina Gandras, and Maarten De Vos (2012). "How about taking a low-cost, small, and wireless EEG for a walk?" In: *Psychophysiology* 49.11, pp. 1617–1621 (cit. on p. 24).

Dehais, Frédéric, Alban Duprès, Sarah Blum, Nicolas Drougard, Sébastien Scannella, Raphaëlle N. Roy, and Fabien Lotte (2019). "Monitoring Pilot's Mental Workload Using ERPs and Spectral Power with a Six-Dry-Electrode EEG System in Real Flight Conditions". In: *Sensors* 19.6. DOI: 10.3390/s19061324 (cit. on p. 26).

Delmas, Jean-Pierre, Mohammed Nabil El Korso, Stefano Fortunati, and Frédéric Pascal (2024). *Elliptically symmetric distributions in signal processing and machine learning*. Vol. 1. Springer (cit. on p. 81).

Delorme, Arnaud (2023). "EEG is better left alone". In: *Scientific reports* 13.1, p. 2372 (cit. on p. 19).

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society: series B (methodological)* 39.1, pp. 1–22 (cit. on p. 186).

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "Imagenet: A large-scale hierarchical image database". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, pp. 248–255 (cit. on p. 158).

Derrow-Pinion, Austin, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, et al. (2021). "Eta prediction with graph neural networks in Google Maps". In: *Proceedings of the 30th ACM international conference on information & knowledge management*, pp. 3767–3776 (cit. on p. 3).

Diederik, P Kingma and Welling Max (2019). "An introduction to variational autoencoders". In: *Foundations and Trends® in Machine Learning* 12.4, pp. 307–392 (cit. on p. 94).

Diepeveen, Willem, Georgios Batzolis, Zakhar Shumaylov, and Carola-Bibiane Schönlieb (2025). "Score-based Pullback Riemannian Geometry: Extracting the Data Manifold Geometry using Anisotropic Flows". In: *Forty-second International Conference on Machine Learning* (cit. on p. 89).

Ding, Jiarui and Aviv Regev (2021). "Deep generative model embedding of single-cell RNA-Seq profiles on hyperspheres and hyperbolic spaces". In: *Nature communications* 12.1, p. 2554 (cit. on p. 55).

Donoho, David L. and Carrie Grimes (May 13, 2003). "Hessian Eigenmaps: Locally Linear Embedding Techniques for High-Dimensional Data". In: *Proceedings of the National Academy of Sciences* 100.10, pp. 5591–5596. DOI: 10.1073/pnas.1031596100 (cit. on p. 49).

Downs, Thomas D. (1972). "Orientation Statistics". In: *Biometrika* 59.3, pp. 665–676 (cit. on pp. 53, 58).

Dryden, Ian L, Alexey Koloydenko, and Diwei Zhou (2009). "Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging". In: *The Annals of Applied Statistics*, pp. 1102–1123 (cit. on pp. 34, 40).

Dryden, Ian L, Xavier Pennec, and Jean-Marc Peyrat (2010). "Power Euclidean metrics for covariance matrices with application to diffusion tensor imaging". In: *arXiv preprint arXiv:1009.3045* (cit. on p. 43).

Duan, Xu, Songyun Xie, Xinzhou Xie, Klaus Obermayer, Yujie Cui, and Zhenzhen Wang (2021). "An Online Data Visualization Feedback Protocol for Motor Imagery-Based BCI Training". In: *Frontiers in Human Neuroscience* Volume 15 - 2021. DOI: 10.3389/fnhum.2021.625983 (cit. on p. 185).

Dudoit, Sandrine, Jane Fridlyand, and Terence P Speed (2002). "Comparison of Discrimination Methods for the Classification of Tumors Using Gene Expression Data". In: *Journal of*

the American Statistical Association 97.457, pp. 77–87. DOI: 10.1198/016214502753479248 (cit. on p. 97).

Eaton, M.L. (2007). *Multivariate Statistics: A Vector Space Approach*. Institute of Mathematical Statistics. Lecture notes-monograph series. Institute of Mathematical Statistics (cit. on pp. 42, 86).

El Ouahidi, Yassine, Jonathan Lys, Philipp Thölke, Nicolas Farrugia, Bastien Pasdeloup, Vincent Gripon, Karim Jerbi, and Giulia Lioi (2025). "REVE: A Foundation Model for EEG: Adapting to Any Setup with Large-Scale Pretraining on 25,000 Subjects". In: *Advances in Neural Information Processing Systems* (cit. on pp. 22, 23, 26, 188).

Émery, Michel and Gabriel Mokobodzki (1991). "Sur le barycentre d'une probabilité dans une variété". In: *Séminaire de probabilités XXV*. Ed. by Jacques Azéma, Paul A. Meyer, and Marc Yor. Vol. 1485. Lecture Notes in Mathematics. Springer, pp. 220–233. DOI: 10.1007/BFb0100858 (cit. on p. 72).

Engemann, Denis and Alexandre Gramfort (2015). "Automated model selection in covariance estimation and spatial whitening of MEG and EEG signals". In: *NeuroImage* 108, pp. 328–342. DOI: https://doi.org/10.1016/j.neuroimage.2014.12.040 (cit. on p. 28).

Esser, Patrick, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. (2024). "Scaling rectified flow transformers for high-resolution image synthesis". In: *Forty-first international conference on machine learning* (cit. on pp. 112, 252).

Evans, Merran Gael Anderson, Nicholas James Hastings, and Brian Peacock (2001). "Statistical Distributions, Third Edition". In: *Measurement Science and Technology* 12, pp. 117–117 (cit. on p. 73).

Fairclough, Stephen H and Fabien Lotte (2020). "Grand challenges in neurotechnology and system neuroergonomics". In: *Frontiers in Neuroergonomics* 1, p. 602504 (cit. on pp. 4, 25).

Faller, Josef, Carmen Vidaurre, Teodoro Solis-Escalante, Christa Neuper, and Reinhold Scherer (2012). "Autocalibration and Recurrent Adaptation: Towards a Plug and Play Online ERD-BCI". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 20.3, pp. 313–319. DOI: 10.1109/TNSRE.2012.2189584 (cit. on pp. 108, 125, 238).

Falorsi, Luca, Pim De Haan, Tim R Davidson, and Patrick Forré (2019). "Reparameterizing distributions on Lie groups". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 3244–3253 (cit. on p. 54).

Farwell, Lawrence Ashley and Emanuel Donchin (1988). "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials". In: *Electroencephalography and clinical Neurophysiology* 70.6, pp. 510–523 (cit. on p. 19).

Fatourechi, Mehrdad, Ali Bashashati, Rabab K Ward, and Gary E Birch (2007). "EMG and EOG artifacts in Brain-Computer interface systems: A survey". In: *Clinical neurophysiology* 118.3, pp. 480–494 (cit. on pp. 148, 258).

Fazel-Rezai, Reza, Brendan Z. Allison, Christoph Guger, Eric W. Sellers, Sonja C. Kleih, and Andrea Kübler (2012). "P300 Brain-Computer interface: current challenges and emerging trends". In: *Frontiers in Neuroengineering* Volume 5 - 2012. DOI: 10.3389/fneng.2012.00014 (cit. on p. 19).

Feragen, Aasa, Francois Lauze, and Soren Hauberg (2015). "Geodesic exponential kernels: When curvature and linearity conflict". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3032–3042 (cit. on pp. 48, 58, 106, 251).

Fisher, Aaron, Cynthia Rudin, and Francesca Dominici (2019). "All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously". In: *Journal of Machine Learning Research* 20.177, pp. 1–81 (cit. on pp. 166, 260).

Fisher, Ronald Aylmer (1953). "Dispersion on a sphere". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 217.1130, pp. 295–305 (cit. on pp. 51, 52, 55, 187).

Flamary, Rémi, Cédric Vincent-Cuaz, Nicolas Courty, Alexandre Gramfort, Oleksii Kachaiev, Huy Quang Tran, Laurène David, Clément Bonet, Nathan Cassereau, Théo Gnassounou, Eloi Tanguy, Julie Delon, Antoine Collas, Sonia Mazelet, Laetitia Chapel, Tanguy Kerdoncuff, Xizheng Yu, Matthew Feickert, Paul Krzakala, Tianlin Liu, and Eduardo Fernandes Montesuma (2024). *POT Python Optimal Transport (version 0.9.5)* (cit. on p. 123).

Fletcher, P. Thomas and Sarang Joshi (Feb. 2007). "Riemannian geometry for the statistical analysis of diffusion tensor data". In: *Signal Process.* 87.2, pp. 250–262. DOI: 10.1016/j.sigpro.2005.12.018 (cit. on p. 49).

Fletcher, P.T., Conglin Lu, S.M. Pizer, and Sarang Joshi (Aug. 2004). "Principal Geodesic Analysis for the Study of Nonlinear Statistics of Shape". In: *IEEE Transactions on Medical Imaging* 23.8, pp. 995–1005. DOI: 10.1109/TMI.2004.831793 (cit. on pp. 49, 58, 157, 158).

Fletcher, Thomas (2011). "Geodesic regression on Riemannian manifolds". In: *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability*, pp. 75–86 (cit. on pp. 47, 58, 59).

Förstner, Wolfgang and Boudewijn Moonen (2003). "A Metric for Covariance Matrices". In: *Geodesy-The Challenge of the 3rd Millennium*. Ed. by Erik W. Grafarend, Friedrich W. Krumm, and Volker S. Schwarze. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 299–309. DOI: 10.1007/978-3-662-05296-9_31 (cit. on p. 34).

Fréchet, Maurice (1948). "Les éléments aléatoires de nature quelconque dans un espace distancié". fr. In: *Annales de l'institut Henri Poincaré* 10.4, pp. 215–310 (cit. on pp. 38, 82, 85).

Friedl, Katharina, Noémie Jaquier, Alyx Liao, and Danica Kragic (2025). "Learning Hamiltonian dynamics at scale: A differential-geometric approach". In: *arXiv preprint arXiv:2509.24627* (cit. on p. 4).

Friedl, Katharina, Noémie Jaquier, Jens Lundell, Tamim Asfour, and Danica Kragic (2025). "A riemannian framework for learning reduced-order lagrangian dynamics". In: *International Conference on Learning Representations*. Vol. 2025, pp. 57775–57802 (cit. on p. 4).

Fuchs, Fabian, Daniel Worrall, Volker Fischer, and Max Welling (2020). "Se (3)-transformers: 3d roto-translation equivariant attention networks". In: *Advances in neural information processing systems* 33, pp. 1970–1981 (cit. on p. 58).

Galaz-Garcia, Fernando, Marios Papamichalis, Kathryn Turnbull, Simon Lunagomez, and Edoardo Airoldi (2022). "Wrapped distributions on homogeneous Riemannian manifolds". In: *arXiv preprint arXiv:2204.09790* (cit. on pp. 55, 58, 67, 80, 83, 138, 175, 248).

Gallet, Matthieu, Florent Bouchard, Ammar Mian, and Guillaume Ginolhac (Feb. 2026). "ARMAGNAC: A New Parametric Batch Normalization Layer for SPDNet Architecture". working paper or preprint (cit. on p. 57).

Ganea, Octavian, Gary Bécigneul, and Thomas Hofmann (2018). "Hyperbolic neural networks". In: *Advances in neural information processing systems* 31 (cit. on pp. 47, 57).

Gao, Ruiqi, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin P. Murphy, and Tim Salimans (2024). *Diffusion Meets Flow Matching: Two Sides of the Same Coin*. URL: https://diffusionflow.github.io/ (cit. on p. 112).

Garcia-Hernando, Guillermo, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim (2018). "First-person hand action benchmark with rgb-d videos and 3d hand pose annotations". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 409–419 (cit. on p. 159).

Garrett, Deon, David A Peterson, Charles W Anderson, and Michael H Thaut (2003). "Comparison of linear, nonlinear, and feature selection methods for EEG signal classification".

In: *IEEE Transactions on neural systems and rehabilitation engineering* 11.2, pp. 141–144 (cit. on pp. 22, 23).

Gartner, Thomas (2008). *Kernels for structured data*. Vol. 72. World Scientific (cit. on p. 106).

Gärtner, Thomas, John W. Lloyd, and Peter A. Flach (2002). "Kernels for structured data". In: *Proceedings of the 12th International Conference on Inductive Logic Programming*. ILP'02. Sydney, Australia: Springer-Verlag, pp. 66–83 (cit. on p. 47).

Gelman, Andrew, John B Carlin, Hal S Stern, and Donald B Rubin (1995). *Bayesian data analysis*. chapman and hall/cRc (cit. on p. 183).

Gerdes, Mathis, Pim de Haan, Corrado Rainone, Roberto Bondesan, and Miranda CN Cheng (2023). "Learning lattice quantum field theories with equivariant continuous flows". In: *SciPost Physics* 15.6, p. 238 (cit. on p. 5).

Ghojogh, Benyamin, Ali Ghodsi, Fakhri Karray, and Mark Crowley (2021). "Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey". In: *arXiv preprint arXiv:2101.00734* (cit. on p. 186).

Gifford, Alessandro T., Kshitij Dwivedi, Gemma Roig, and Radoslaw M. Cichy (2022). "A large and rich EEG dataset for modeling human visual object recognition". In: *NeuroImage* 264, p. 119754. DOI: https://doi.org/10.1016/j.neuroimage.2022.119754 (cit. on p. 26).

Gnassounou, Théo, Antoine Collas, Rémi Flamary, and Alexandre Gramfort (2026). "PSDNorm: Test-Time Temporal Normalization for Deep Learning in Sleep Staging". In: *International Conference on Learning Representations (ICLR)* (cit. on p. 24).

Goh, Alvina, Christophe Lenglet, Paul M. Thompson, and René Vidal (2011). "A nonparametric Riemannian framework for processing high angular resolution diffusion images and its applications to ODF-based morphometry". In: *NeuroImage* 56.3, pp. 1181–1201. DOI: 10.1016/j.neuroimage.2011.01.053 (cit. on p. 57).

Goh, Alvina and Rene Vidal (2008). "Clustering and dimensionality reduction on Riemannian manifolds". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–7. DOI: 10.1109/CVPR.2008.4587422 (cit. on pp. 49, 58).

Goldstine, Herman H and John Von Neumann (1951). "Numerical inverting of matrices of high order. II". In: *Proceedings of the American Mathematical Society* 2.2, pp. 188–202 (cit. on p. 123).

Golub, G.H. and C.F. Van Loan (2013). *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press (cit. on pp. 38, 39, 236).

González-Duque, Miguel, Leonel Rozo, Noémie Jaquier, and Søren Hauberg (2025). "Riemann2: Learning Riemannian Submanifolds from Riemannian Data". In: *International Conference on Artifi-cial Intelligence and Statistics (AISTATS)*. Vol. 258 (cit. on pp. 55, 89).

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press (cit. on p. 185).

Gozard, I. (2009). *Théorie de Galois*. Mathématiques à l'université. Ellipses (cit. on p. 156).

Gramann, Klaus, Joseph T Gwin, Nima Bigdely-Shamlo, Daniel P Ferris, and Scott Makeig (2010). "Visual evoked responses during standing and walking". In: *Frontiers in human neuroscience* 4, p. 202 (cit. on p. 24).

Gray, Robert M (2009). *Probability, random processes, and ergodic properties*. Vol. 1. Springer (cit. on p. 27).

Grosse-Wentrup, Moritz and Bernhard Schölkopf (2012). "High gamma-power predicts performance in sensorimotor-rhythm brain–computer interfaces". In: *Journal of neural engineering* 9.4, p. 046001 (cit. on p. 26).

— (2013). "A Review of Performance Variations in SMR-Based Brain-Computer Interfaces (BCIs)". In: *Brain-Computer Interface Research: A State-of-the-Art Summary*. Ed. by Christoph Guger, Brendan Z. Allison, and Günter Edlinger. SpringerBriefs in Electrical and Computer Engineering. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-642-36083-1_5 (cit. on pp. 4, 25, 26).

Grötschel, Martin, László Lovász, and Alexander Schrijver (2012). *Geometric algorithms and combinatorial optimization*. Vol. 2. Springer Science & Business Media (cit. on p. 33).

Guo, Yunhui, Haoran Guo, and Stella X Yu (2022). "Co-sne: Dimensionality reduction and visualization for hyperbolic data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21–30 (cit. on pp. 50, 58).

Hajri, Hatem, Ioana Ilea, Salem Said, Lionel Bombrun, and Yannick Berthoumieu (2016). "Riemannian Laplace distribution on the space of symmetric positive definite matrices". In: *Entropy* 18.3, p. 98 (cit. on p. 54).

Häkkinen, Antti, Juha Koiranen, Julia Casado, Katja Kaipio, Oskari Lehtonen, Eleonora Petrucci, Johanna Hynninen, Sakari Hietanen, Olli Carpen, Luca Pasquini, et al. (2020). "qSNE: quadratic rate t-SNE optimizer with automatic parameter tuning for large datasets". In: *Bioinformatics* 36.20, pp. 5086–5092 (cit. on pp. 157, 158).

Halder, Sebastian, Balint Varkuti, Martin Bogdan, Andrea Kübler, Wolfgang Rosenstiel, Ranganatha Sitaram, and Niels Birbaumer (2013). "Prediction of brain-computer interface aptitude from individual brain structure". In: *Frontiers in human neuroscience* 7, p. 105 (cit. on p. 26).

Hammer, Eva Maria, Sebastian Halder, Benjamin Blankertz, Claudia Sannelli, Thorsten Dickhaus, Sonja Kleih, Klaus-Robert Müller, and Andrea Kübler (2012). "Psychological predictors of SMR-BCI performance". In: *Biological psychology* 89.1, pp. 80–86 (cit. on p. 26).

Hanik, Martin, Hans-Christian Hege, Anja Hennemuth, and Christoph von Tycowicz (2020). "Nonlinear regression on manifolds for shape analysis using intrinsic Bézier splines". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, pp. 617–626 (cit. on p. 47).

Harandi, Mehrtash T, Mathieu Salzmann, and Richard Hartley (2014). "From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices". In: *European conference on computer vision*. Springer, pp. 17–32 (cit. on pp. 50, 56, 58).

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant (Sept. 2020). "Array programming with NumPy". In: *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2 (cit. on pp. 38, 112).

Hartmann, Kay Gregor, Robin Tibor Schirrmeister, and Tonio Ball (2018). "EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals". In: *arXiv preprint arXiv:1806.01875* (cit. on p. 26).

Haseeb, Muhammad, Rashid Nadeem, Nazia Sultana, Noman Naseer, Hammad Nazeer, and Frédéric Dehais (2025). "Monitoring pilots' mental workload in real flight conditions using multinomial logistic regression with a ridge estimator". In: *Frontiers in Robotics and AI* 12. DOI: 10.3389/frobt.2025.1441801 (cit. on p. 24).

Haselsteiner, Ernst and Gert Pfurtscheller (2000). "Using time-dependent neural networks for EEG classification". In: *IEEE transactions on rehabilitation engineering* 8.4, pp. 457–463 (cit. on p. 22).

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer. DOI: 10.1007/978-0-387-84858-7 (cit. on pp. 6, 46, 47, 93, 97, 108, 244, 250).

Hauberg, Søren (July 2018). "Directional Statistics with the Spherical Normal Distribution". In: *2018 21st International Conference on Information Fusion (FUSION)*, pp. 704–711. DOI: 10.23919/ICIF.2018.8455242 (cit. on p. 52).

Haufe, Stefan, Frank Meinecke, Kai Görgen, Sven Dähne, John-Dylan Haynes, Benjamin Blankertz, and Felix Bießmann (2014). "On the interpretation of weight vectors of linear models in multivariate neuroimaging". In: *NeuroImage* 87, pp. 96–110. DOI: https://doi.org/10.1016/j.neuroimage.2013.10.067 (cit. on pp. 21, 150, 182).

Helgason, S. (1979). *Differential Geometry, Lie Groups, and Symmetric Spaces*. Pure and Applied Mathematics. Academic Press (cit. on p. 90).

Herman, Pawel, Girijesh Prasad, Thomas Martin McGinnity, and Damien Coyle (2008). "Comparative analysis of spectral approaches to feature extraction for EEG-based motor imagery classification". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 16.4, pp. 317–326 (cit. on p. 21).

Higham, Nicholas J. (Jan. 2008). *Functions of Matrices*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898717778 (cit. on pp. 118, 121).

Hinkle, Jacob, Prasanna Muralidharan, P Thomas Fletcher, and Sarang Joshi (2012). "Polynomial regression on Riemannian manifolds". In: *European conference on computer vision*. Springer, pp. 1–14 (cit. on p. 47).

Hinton, Geoffrey and Sam Roweis (2002). "Stochastic neighbor embedding". In: *Advances in neural information processing systems* 15 (cit. on p. 49).

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33, pp. 6840–6851 (cit. on pp. 1, 112, 251).

Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. (2022). "Training compute-optimal large language models". In: *arXiv preprint arXiv:2203.15556* (cit. on p. 188).

Hofmann, Thomas, Bernhard Schölkopf, and Alexander J Smola (2008). "Kernel methods in machine learning". In: *The Annals of Statistics* (cit. on pp. 1, 47, 106).

Horev, Inbal, Florian Yger, and Masashi Sugiyama (Nov. 2016). "Geometry-Aware Principal Component Analysis for Symmetric Positive Definite Matrices". In: *Asian Conference on Machine Learning*. Ed. by Geoffrey Holmes and Tie-Yan Liu. Vol. 45. Proceedings of Machine Learning Research. Hong Kong: PMLR, pp. 1–16 (cit. on pp. 49, 50, 56, 58, 158).

Horn, Roger A. and Charles R. Johnson (1994). *Topics in Matrix Analysis*. Cambridge; New York: Cambridge University Press (cit. on p. 156).

— (2012). *Matrix Analysis*. 2nd ed. Cambridge: Cambridge University Press (cit. on pp. 31, 33).

Hosseini, Mohammad-Parsa, Hamid Soltanian-Zadeh, Kost Elisevich, and Dario Pompili (2016). "Cloud-based deep learning of big EEG data for epileptic seizure prediction". In: *2016 IEEE global conference on signal and information processing (GlobalSIP)*. IEEE, pp. 1151–1155 (cit. on p. 24).

Hotelling, Harold (1933). "Analysis of a complex of statistical variables into principal components." In: *Journal of educational psychology* 24.6, p. 417 (cit. on p. 49).

— (1992). "The generalization of Student's ratio". In: *Breakthroughs in statistics: Foundations and basic theory*. Springer, pp. 54–65 (cit. on p. 184).

Hua, Jinxi, Yuanxun Zhang, Benjamin de Foy, Xiaodong Mei, Jing Shang, and Chuan Feng (Jan. 2021). "Competing PM2.5 and NO2 Holiday Effects in the Beijing Area Vary Locally Due to Differences in Residential Coal Burning and Traffic Patterns". In: *Science of The Total Environment* 750, p. 141575. DOI: 10.1016/j.scitotenv.2020.141575 (cit. on p. 102).

Huang, Zhiwu and Luc Van Gool (2017). "A riemannian network for spd matrix learning". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1 (cit. on pp. 10, 23, 56–58, 170).

Huang, Zhiwu, Chengde Wan, Thomas Probst, and Luc Van Gool (2017). "Deep learning on lie groups for skeleton-based action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6099–6108 (cit. on p. 58).

Huang, Zhiwu, Ruiping Wang, Shiguang Shan, and Xilin Chen (2015). "Projection Metric Learning on Grassmann Manifold with Application to Video based Face Recognition". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 140–149. DOI: 10.1109/CVPR.2015.7298609 (cit. on p. 57).

Huang, Zhiwu, Ruiping Wang, Shiguang Shan, Xianqiu Li, and Xilin Chen (2015). "Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification". In: *International conference on machine learning*. PMLR, pp. 720–729 (cit. on p. 180).

Huang, Zhiwu, Jiqing Wu, and Luc Van Gool (2018). "Building deep networks on grassmann manifolds". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1 (cit. on p. 57).

Huber, Peter J. (1981). *Robust Statistics*. Wiley Series in Probability and Statistics. Wiley (cit. on p. 29).

Hyvärinen, A. and E. Oja (2000). "Independent component analysis: algorithms and applications". In: *Neural Networks* 13.4, pp. 411–430. DOI: 10.1016/S0893-6080(00)00026-5 (cit. on pp. 19, 104).

Igashov, Ilia, Hannes Stärk, Clément Vignac, Arne Schneuing, Victor Garcia Satorras, Pascal Frossard, Max Welling, Michael Bronstein, and Bruno Correia (2024). "Equivariant 3D-conditional diffusion model for molecular linker design". In: *Nature Machine Intelligence* 6.4, pp. 417–427 (cit. on p. 5).

Ilyas, Andrew, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Mądry (2019). "Adversarial examples are not bugs, they are features". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc. (cit. on p. 171).

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr, pp. 448–456 (cit. on p. 57).

Iruthayarajan, M Willjuice and S Baskar (2010). "Covariance matrix adaptation evolution strategy based design of centralized PID controller". In: *Expert systems with Applications* 37.8, pp. 5775–5781 (cit. on p. 4).

Jaquier, Noémie and Sylvain Calinon (2017). "Gaussian mixture regression on symmetric positive definite matrices manifolds: Application to wrist motion estimation with sEMG". In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 59–64 (cit. on p. 54).

Jaquier, Noémie and Leonel Rozo, eds. (2027). *Riemannian Manifolds in Robot Learning, Optimization, and Control*. Cambridge University Press (cit. on pp. 4, 179, 180, 187).

Jaquier, Noémie, Leonel Rozo, and Tamim Asfour (2024). "Unraveling the Single Tangent Space Fallacy: An Analysis and Clarification for Applying Riemannian Geometry in Robot Learning". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 242–249. DOI: 10.1109/ICRA57147.2024.10611701 (cit. on pp. 59, 107, 120, 179, 180, 253).

Jayasumana, Sadeep, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi (2013). "Kernel methods on the Riemannian manifold of symmetric positive definite matrices". In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 73–80 (cit. on pp. 48, 58, 107, 251).

Jeffreys, Harold (1946). "An Invariant Form for the Prior Probability in Estimation Problems". In: *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* 186.1007, pp. 453–461 (cit. on pp. 44, 184).

Jenkovszky, László, Matthew J Lake, and Vladimir Soloviev (2023). *János Bolyai, Carl Friedrich Gauss, Nikolai Lobachevsky and the New Geometry: Foreword* (cit. on pp. 2, 243).

Jeunet, Camille, Denis Hauw, and Jose del R. Millán (2020). "Sport Psychology: Technologies Ahead". In: *Frontiers in Sports and Active Living* Volume 2 - 2020. DOI: 10.3389/fspor.2020.00010 (cit. on p. 24).

Jeunet, Camille, Bernard N'Kaoua, and Fabien Lotte (2016). "Advances in user-training for mental-imagery-based BCI control: Psychological and cognitive factors and their neural correlates". In: *Progress in brain research* 228, pp. 3–35 (cit. on p. 26).

Jiang, Wei-Bang, Liming Zhao, and Bao-Liang Lu (2024). "Large brain model for learning generic representations with tremendous EEG data in BCI". In: *International Conference on Learning Representations*. Vol. 2024, pp. 16405–16426 (cit. on pp. 22, 23, 188).

Jiang, Yangdi, Xiaotian Chang, Lei Ding, Linglong Kong, and Bei Jiang (2025). *Exponential-Wrapped Mechanisms for Differential Privacy on Hadamard Manifolds* (cit. on p. 55).

Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics. New York: Springer-Verlag. DOI: 10.1007/b98835 (cit. on p. 46).

Ju, Ce and Cuntai Guan (2023). "Graph neural networks on SPD manifolds for motor imagery classification: A perspective from the time–frequency analysis". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.12, pp. 17701–17715 (cit. on p. 57).

Jumper, John, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. (2021). "Highly accurate protein structure prediction with AlphaFold". In: *nature* 596.7873, pp. 583–589 (cit. on pp. 3, 187).

Kagan, A.M., I.U.V. Linnik, and C.R. Rao (1973). *Characterization Problems in Mathematical Statistics*. Probability and Statistics Series. Wiley (cit. on p. 51).

Kaifosh, Patrick and Thomas R Reardon (2025). "A generic non-invasive neuromotor interface for human-computer interaction". In: *Nature* 645.8081, pp. 702–711 (cit. on pp. 22, 188).

Kalunga, Emmanuel, Sylvain Chevallier, and Quentin Barthélemy (2015). "Data augmentation in Riemannian space for brain-computer interfaces". In: *STAMLINS* (cit. on p. 26).

Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei (2020). "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (cit. on pp. 22, 188).

Karcher, H. (1977). "Riemannian center of mass and mollifier smoothing". In: *Communications on Pure and Applied Mathematics* 30.5, pp. 509–541. DOI: https://doi.org/10.1002/cpa.3160300502 (cit. on p. 38).

Kasahara, Kazumi, Charles Sayo DaSalla, Manabu Honda, and Takashi Hanakawa (2015). "Neuroanatomical correlates of brain–computer interface performance". In: *NeuroImage* 110, pp. 95–100. DOI: 10.1016/j.neuroimage.2015.01.055 (cit. on p. 26).

Kato, S. and P. McCullagh (Nov. 2020). "Some properties of a Cauchy family on the sphere derived from the Möbius transformations". In: *Bernoulli* 26.4, pp. 3224–3248. DOI: 10.3150/20-bej1222 (cit. on p. 54).

Kauffmann, Jacob, Jonas Dippel, Lukas Ruff, Wojciech Samek, Klaus-Robert Müller, and Grégoire Montavon (Mar. 2025). "Explainable AI reveals Clever Hans effects in unsupervised learning models". In: *Nature Machine Intelligence*. DOI: 10.1038/s42256-025-01000-2 (cit. on p. 150).

Kay, S. M. (1997). *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall (cit. on p. 184).

Kendall, David G (1977). "The diffusion of shape". In: *Advances in applied probability* 9.3, pp. 428–430 (cit. on p. 187).

— (1984). "Shape manifolds, procrustean metrics, and complex projective spaces". In: *Bulletin of the London mathematical society* 16.2, pp. 81–121 (cit. on p. 187).

Kent, John T. (1982). "The Fisher-Bingham Distribution on the Sphere". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 44.1, pp. 71–80. DOI: 10.1111/j.2517-6161.1982.tb01189.x (cit. on p. 52).

Klambauer, Günter, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter (2017). "Self-normalizing neural networks". In: *Advances in neural information processing systems* 30 (cit. on p. 236).

Klimovskaia, Anna, David Lopez-Paz, Léon Bottou, and Maximilian Nickel (2020). "Poincaré maps for analyzing complex hierarchies in single-cell data". In: *Nature communications* 11.1, p. 2966 (cit. on pp. 50, 58).

Kobak, Dmitry and Philipp Berens (2019). "The art of using t-SNE for single-cell transcriptomics". In: *Nature communications* 10.1, p. 5416 (cit. on pp. 151, 157, 163, 259).

Kobler, Reinmar, Jun-ichiro Hirayama, Qibin Zhao, and Motoaki Kawanabe (2022). "SPD domain-specific batch normalization to crack interpretable unsupervised domain adaptation in EEG". In: *Advances in Neural Information Processing Systems* 35, pp. 6219–6235 (cit. on pp. 57, 58).

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86 (cit. on p. 43).

Kungurtsev, Vyacheslav, Francesco Rinaldi, and Damiano Zeffiro (2024). "Retraction-based direct search methods for derivative free Riemannian optimization". In: *Journal of Optimization Theory and Applications* 203.2, pp. 1710–1735 (cit. on p. 171).

Kynkäänniemi, Tuomas, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila (2019). "Improved precision and recall metric for assessing generative models". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc. (cit. on p. 125).

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert (2015). "Numba: a LLVM-based Python JIT compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM '15. Austin, Texas: Association for Computing Machinery. DOI: 10.1145/2833157.2833162 (cit. on p. 156).

Lapuschkin, Sebastian, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller (2019). "Unmasking Clever Hans predictors and assessing what machines really learn". In: *Nature communications* 10.1, p. 1096 (cit. on pp. 166, 260).

Lawhern, Vernon J, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance (2018). "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces". In: *Journal of neural engineering* 15.5, p. 056013 (cit. on pp. 22, 23).

Lawley, Derrick Norman and Albert Ernest Maxwell (1962). "Factor analysis as a statistical method". In: *Journal of the Royal Statistical Society. Series D (The Statistician)* 12.3, pp. 209–229 (cit. on p. 186).

Le, Matthew, Apoorv Vyas, Bowen Shi, Brian Karrer, Leda Sari, Rashel Moritz, Mary Williamson, Vimal Manohar, Yossi Adi, Jay Mahadeokar, et al. (2023). "Voicebox: Text-guided multilingual universal speech generation at scale". In: *Advances in neural information processing systems* 36, pp. 14005–14034 (cit. on pp. 112, 252).

Le Bihan, Denis, Jean-François Mangin, Cyril Poupon, Chris A Clark, Sabina Pappata, Nicolas Molko, and Hughes Chabriat (2001). "Diffusion tensor imaging: concepts and applications". In: *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine* 13.4, pp. 534–546 (cit. on p. 4).

Lecaignard, Francoise and J Mattout (2015). "Forward models for EEG/MEG". In: *Brain Mapping*, pp. 549–555 (cit. on p. 30).

LeCun, Yann (2022). "A Path Towards Autonomous Machine Intelligence". In: *Open Review* 62.1, pp. 1–62 (cit. on p. 1).

Ledoit, Olivier and Michael Wolf (Feb. 2004). "A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices". In: *Journal of Multivariate Analysis* 88.2, pp. 365–411. DOI: 10.1016/S0047-259X(03)00096-4 (cit. on pp. 26, 29, 101, 124, 137).

Lee, John M. (2012). *Introduction to Smooth Manifolds*. Vol. 218. Graduate Texts in Mathematics. New York, NY: Springer. DOI: 10.1007/978-1-4419-9982-5 (cit. on pp. 34, 89).

— (2018). *Introduction to Riemannian Manifolds*. Vol. 176. Graduate Texts in Mathematics. Cham: Springer International Publishing. DOI: 10.1007/978-3-319-91755-9 (cit. on pp. 34, 37, 38, 47, 68, 79, 87, 89, 225, 249).

Leeb, Robert, Felix Lee, Claudia Keinrath, Reinhold Scherer, Horst Bischof, and Gert Pfurtscheller (2007). "Brain–Computer Communication: Motivation, Aim, and Impact of Exploring a Virtual Apartment". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 15.4, pp. 473–482. DOI: 10.1109/TNSRE.2007.906956 (cit. on pp. 101, 108).

Leslie, Kate, Matthew TV Chan, Paul S Myles, Andrew Forbes, and Timothy J McCulloch (2010). "Posttraumatic stress disorder in aware patients from the B-aware trial". In: *Anesthesia & analgesia* 110.3, pp. 823–828 (cit. on p. 135).

Lin, Jianhua (1991). "Divergence measures based on the Shannon entropy". In: *IEEE Transactions on Information theory* 37.1, pp. 145–151 (cit. on p. 44).

Lin, Zhenhua (2019). "Riemannian Geometry of Symmetric Positive Definite Matrices via Cholesky Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 40.4, pp. 1353–1370. DOI: 10.1137/18M1221084 (cit. on pp. 34, 40, 245).

Linderman, George C, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger (2019). "Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data". In: *Nature methods* 16.3, pp. 243–245 (cit. on p. 157).

Lipman, Yaron, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le (Feb. 2023). *Flow Matching for Generative Modeling*. DOI: 10.48550/arXiv.2210.02747 (cit. on pp. 1, 94, 110, 112, 114–118, 120, 177, 251, 252).

Lipman, Yaron, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat (Dec. 9, 2024). *Flow Matching Guide and Code*. URL: http://arxiv.org/abs/2412.06264 (visited on 12/13/2024). Pre-published (cit. on pp. 114, 115).

Lipton, Zachary C (2018). "The mythos of model interpretability". In: *Communications of the ACM* 61.10, pp. 36–43 (cit. on pp. 148, 182).

Liu, Jiaqi, Xueling Zhou, Meilin Wang, Ziqi Guo, and Bingo Wing-Kuen Ling (2026). "Lightweight Seizure Detection for Wearable Electronics via Riemannian t-SNE". In: *IEEE Transactions on Consumer Electronics* (cit. on pp. 50, 58, 179).

Liu, Qi, Maximilian Nickel, and Douwe Kiela (2019). "Hyperbolic graph neural networks". In: *Advances in neural information processing systems* 32 (cit. on p. 58).

Liu, Xingchao, Chengyue Gong, and Qiang Liu (2023). "Flow straight and fast: Learning to generate and transfer data with rectified flow". In: *International conference on learning representations (ICLR)* (cit. on pp. 112, 115, 128, 251).

Liu, Xiu-Yun, Wen-Long Wang, Miao Liu, Ming-Yi Chen, Tânia Pereira, Desta Yakob Doda, Yu-Feng Ke, Shou-Yan Wang, Dong Wen, Xiao-Guang Tong, et al. (2025). "Recent applications of EEG-based brain-computer-interface in the medical field". In: *Military Medical Research* 12.1, p. 14 (cit. on p. 24).

Loshchilov, Ilya and Frank Hutter (2017). "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (cit. on p. 237).

Lotte, Fabien (2014). "A tutorial on EEG signal-processing techniques for mental-state recognition in brain–computer interfaces". In: *Guide to brain-computer music interfacing*, pp. 133–161 (cit. on p. 22).

Lotte, Fabien, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, Alain Rakotomamonjy, and Florian Yger (2018). "A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update". In: *Journal of neural engineering* 15.3, p. 031005 (cit. on pp. 21–23).

Lotte, Fabien, Marco Congedo, A Lécuyer, F Lamarche, and B Arnaldi (2007). "A review of classification algorithms for EEG-based brain-computer interfaces". In: *Journal of neural engineering* 4.2, R1–R13 (cit. on p. 22).

Lotte, Fabien, Pauline Dreyer, Sébastien Rimbert, David Trocellier, and Marc Welter (2025). "How to build a brain–computer interface from beginning to end". In: *Neural Interfaces*. Ed. by Davide Valeriani and Theresa Vaughan. Elsevier, pp. 115–141. DOI: 10.1016/B978-0-443-24824-5.00004-1 (cit. on pp. 4, 24, 26, 188).

Lotte, Fabien and Cuntai Guan (2011). "Regularizing Common Spatial Patterns to Improve BCI Designs: Unified Theory and New Algorithms". In: *IEEE Transactions on Biomedical Engineering* 58.2, pp. 355–362 (cit. on p. 21).

Lotte, Fabien, Florian Larrue, and Christian Mühl (2013). "Flaws in current human training protocols for spontaneous brain-computer interfaces: lessons learned from instructional design". In: *Frontiers in human neuroscience* 7, p. 568 (cit. on p. 185).

Lundberg, Scott M and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. Curran Associates, Inc., pp. 4768–4777 (cit. on p. 169).

Ma, Hui and Kai Chen (2024). "Online t-SNE for single-cell RNA-seq". In: *arXiv preprint arXiv:2406.14842* (cit. on pp. 154, 172, 185).

Maaten, Laurens van der (2014). "Accelerating t-SNE using Tree-Based Algorithms". In: *Journal of Machine Learning Research* 15.93, pp. 3221–3245 (cit. on p. 157).

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data Using T-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605 (cit. on pp. 9, 25, 46, 49, 110, 150–153, 157, 163, 177, 185, 239, 258, 259).

Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu (2017). "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (cit. on p. 171).

Makadia, Ameesh, Christopher Geyer, and Kostas Daniilidis (2007). "Correspondence-free structure from motion". In: *International Journal of Computer Vision* 75.3, pp. 311–327 (cit. on p. 5).

Malagò, Luigi, Luigi Montrucchio, and Giovanni Pistone (2018). "Wasserstein Riemannian geometry of Gaussian densities". In: *Information Geometry* 1.2, pp. 137–179 (cit. on pp. 41, 245).

Mallasto, Anton and Aasa Feragen (2018). "Wrapped Gaussian process regression on Riemannian manifolds". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5580–5588 (cit. on p. 55).

Marco, Alonso, Philipp Hennig, Jeannette Bohg, Stefan Schaal, and Sebastian Trimpe (2016). "Automatic LQR tuning based on Gaussian process global optimization". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 270–277 (cit. on p. 4).

Mardia, K.V. (1972). *Statistics of Directional Data*. Probability and Mathematical Statistics a Series of Monographs and Textbooks. Academic Press (cit. on p. 52).

Mardia, K.V. and P.E. Jupp (2009). *Directional Statistics*. Wiley Series in Probability and Statistics. Wiley (cit. on pp. 52–55).

Marissens Cueva, Valérie, Sébastien Rimbert, AM Cebolla Alvarez, Guy Cheron, Claude Meistelman, Seyed Javad Bidgoli, Fabien Lotte, and Laurent Bougrain (2025). "Benchmarking one-class Riemannian EEG classifiers to detect wakefulness under general anesthesia". In: *11th International BCI Meeting* (cit. on pp. 134, 136, 137).

Marissens Cueva, Valérie, Sébastien Rimbert, Ana Maria Cebolla Alvarez, Mathieu Petieau, Iraj Hashemi, Viktoriya Vitkova, Guy Cheron, Claude Meistelman, Philippe Guerci, Denis Schmartz, Seyed Javad Bidgoli, Laurent Bougrain, and Fabien Lotte (2024). "One-Class Riemannian EEG Classifier to Detect Anesthesia". In: *Proc. NEC* (cit. on pp. 137, 255).

Marsaglia, George and Wai Wan Tsang (2000). "The Ziggurat Method for Generating Random Variables". In: *Journal of Statistical Software* 5.8, pp. 1–7. DOI: 10.18637/jss.v005.i08 (cit. on pp. 112, 251).

Massart, Estelle and P-A Absil (2020). "Quotient geometry with simple geodesics for the manifold of fixed-rank positive-semidefinite matrices". In: *SIAM Journal on Matrix Analysis and Applications* 41.1, pp. 171–198 (cit. on p. 186).

Massart, Estelle, Julien M Hendrickx, and P-A Absil (2019). "Curvature of the manifold of fixed-rank positive-semidefinite matrices endowed with the Bures–Wasserstein metric". In: *International Conference on Geometric Science of Information*. Springer, pp. 739–748 (cit. on p. 186).

Mathieu, Emile, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh (2019). "Continuous Hierarchical Representations with Poincaré Variational Auto-Encoders". In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc. (cit. on pp. 54, 58, 89, 229).

Matthews, Gerald, Lauren E Reinerman-Jones, Daniel J Barber, and Julian Abich IV (2015). "The psychometrics of mental workload: Multiple measures are sensitive but divergent". In: *Human factors* 57.1, pp. 125–143 (cit. on p. 24).

McCann, Robert J. (1997). "A Convexity Principle for Interacting Gases". In: *Advances in Mathematics* 128.1, pp. 153–179. DOI: https://doi.org/10.1006/aima.1997.1634 (cit. on p. 116).

McInnes, Leland, John Healy, and James Melville (2018). "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (cit. on pp. 46, 50, 158).

Meilă, Marina and Hanyu Zhang (2024). "Manifold learning: What, how, and why". In: *Annual Review of Statistics and Its Application* 11.1, pp. 393–417 (cit. on p. 46).

Mellot, Apolline, Antoine Collas, Sylvain Chevallier, Denis Engemann, and Alexandre Gramfort (2024). "Physics-informed and unsupervised Riemannian domain adaptation for machine learning on heterogeneous EEG datasets". In: *2024 32nd European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1367–1371 (cit. on p. 19).

Mellot, Apolline, Antoine Collas, Sylvain Chevallier, Alexandre Gramfort, and Denis Engemann (2024). "Geodesic optimization for predictive shift adaptation on EEG data". In: *Advances in neural information processing systems* 37, pp. 32828–32855 (cit. on pp. 47, 58, 59, 179).

Micheli, Alessandro, Yueqi Cao, Anthea Monod, and Samir Bhatt (2026). "Riemannian Neural Optimal Transport". In: *Forty-third International Conference on Machine Learning* (cit. on p. 187).

Millán, J d R, Rüdiger Rupp, Gernot R Müller-Putz, Roderick Murray-Smith, Claudio Giugliemma, Michael Tangermann, Carmen Vidaurre, Febo Cincotti, Andrea Kübler, Robert Leeb, et al. (2010). "Combining brain–computer interfaces and assistive technologies: state-of-the-art and challenges". In: *Frontiers in neuroscience* 4, p. 161 (cit. on p. 16).

Moakher, Maher (Jan. 2005). "A Differential Geometric Approach to the Geometric Mean of Symmetric Positive-Definite Matrices". In: *SIAM Journal on Matrix Analysis and Applications* 26.3, pp. 735–747. DOI: 10.1137/S0895479803436937 (cit. on pp. 38, 82, 96).

Si-Mohammed, Hakim, Jimmy Petit, Camille Jeunet, Ferran Argelaguet, Fabien Spindler, Andéol Evain, Nicolas Roussel, Géry Casiez, and Anatole Lécuyer (2018). "Towards BCI-based interfaces for augmented reality: feasibility, design and evaluation". In: *IEEE transactions on visualization and computer graphics* 26.3, pp. 1608–1621 (cit. on p. 24).

Molnar, Christoph (2025). *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 3rd ed. (cit. on pp. 147, 164, 166, 169, 170, 258, 260).

Muirhead, Robb J. (1982). *Aspects of Multivariate Statistical Theory*. Wiley Series in Probability and Mathematical Statistics. Wiley (cit. on p. 53).

Mulcahy, Karen A and Keith C Clarke (2001). "Symbolization of map projection distortion: a review". In: *Cartography and geographic information science* 28.3, pp. 167–182 (cit. on p. 149).

Mullen, Tim R, Christian AE Kothe, Yu Mike Chi, Alejandro Ojeda, Trevor Kerth, Scott Makeig, Tzyy-Ping Jung, and Gert Cauwenberghs (2015). "Real-time neuroimaging and cognitive monitoring using wearable dry EEG". In: *IEEE transactions on biomedical engineering* 62.11, pp. 2553–2567 (cit. on p. 24).

Muller, Mervin E (1958). "An inverse method for the generation of random normal deviates on large-scale computers". In: *Mathematics of Computation* 12.63, pp. 167–174 (cit. on p. 112).

Murphy, Douglas P., Ou Bai, Ashraf S. Gorgey, John Fox, William T. Lovegreen, Brian W. Burkhardt, Roozbeh Atri, Juan S. Marquez, Qi Li, and Ding-Yu Fei (2017). "Electroencephalogram-Based Brain–Computer Interface and Lower-Limb Prosthesis Control: A Case Study". In: *Frontiers in Neurology* 8 (cit. on p. 23).

Murphy, Kevin P (2007). *Conjugate Bayesian analysis of the Gaussian distribution*. Technical report. University of British Columbia (cit. on p. 183).

Musso, Mariacristina, David Hübner, Sarah Schwarzkopf, Maria Bernodusson, Pierre LeVan, Cornelius Weiller, and Michael Tangermann (2022). "Aphasia recovery by language training using a brain–computer interface: a proof-of-concept study". In: *Brain communications* 4.1, fcac008 (cit. on p. 24).

Naeem, Muhammad Ferjad, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo (2020). "Reliable fidelity and diversity metrics for generative models". In: *International conference on machine learning*. PMLR, pp. 7176–7185 (cit. on pp. 125, 254).

Nagano, Yoshihiro, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama (May 2019). "A Wrapped Normal Distribution on Hyperbolic Space for Gradient-Based Learning". In: *Proceedings of the 36th International Conference on Machine Learning*. PMLR, pp. 4693–4702 (cit. on pp. 54, 58, 67, 80, 89, 138, 175, 229, 248).

Nam, {Chang S.}, Anton Nijholt, and Fabien Lotte, eds. (Jan. 2018). *Brain-Computer Interfaces Handbook: Technological and Theoretical Advances*. English. CRC Press (Taylor & Francis) (cit. on p. 16).

Navarro-Sune, Xavier, AL Hudson, F De Vico Fallani, Jacques Martinerie, Adrien Witon, Pierre Pouget, Mathieu Raux, Thomas Similowski, and Mario Chavez (2016). "Riemannian geometry applied to detection of respiratory states from EEG signals: the basis for a brain–ventilator interface". In: *IEEE Transactions on Biomedical Engineering* 64.5, pp. 1138–1148 (cit. on p. 134).

Nesterov, Yurii and Vladimir Spokoiny (2017). "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2, pp. 527–566 (cit. on p. 171).

Neuper, Christa, Reinhold Scherer, Miriam Reiner, and Gert Pfurtscheller (2005). "Imagery of motor actions: Differential effects of kinesthetic and visual–motor mode of imagery in single-trial EEG". In: *Cognitive brain research* 25.3, pp. 668–677 (cit. on pp. 166, 260).

Neyman, Jerzy and Egon Sharpe Pearson (1992). "On the problem of the most efficient tests of statistical hypotheses". In: *Breakthroughs in Statistics: Foundations and Basic Theory*. Springer, pp. 73–108 (cit. on p. 185).

Nguyen, T Tin, Hien D Nguyen, Faicel Chamroukhi, and Geoffrey J McLachlan (2020). "Approximation by finite mixtures of continuous density functions that vanish at infinity". In: *Cogent Mathematics & Statistics* 7.1, p. 1750861 (cit. on p. 186).

Nguyen, Thu, Rabindra Khadka, Nhan Phan, Anis Yazidi, Pål Halvorsen, and Michael A. Riegler (2023). "Combining datasets to improve model fitting". In: *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9. DOI: 10.1109/IJCNN54540.2023.10191273 (cit. on p. 19).

Nguyen, Xuan Son (Oct. 2022). "A Gyrovector Space Approach for Symmetric Positive Semi-definite Matrix Learning". In: *European Conference on Computer Vision 2022 (ECCV 2022)*. Tel Aviv, Israel (cit. on p. 57).

Nguyen, Xuan Son and Shuo Yang (2023). "Building neural networks on matrix manifolds: A gyrovector space approach". In: *International Conference on Machine Learning*. PMLR, pp. 26031–26062 (cit. on p. 47).

Nguyen, Xuan Son, Shuo Yang, and Aymeric Histace (2024). "Matrix Manifold Neural Networks++". In: *The Twelfth International Conference on Learning Representations* (cit. on pp. 47, 57).

Nielsen, Frank and Sylvain Boltz (2011). "The burbea-rao and bhattacharyya centroids". In: *IEEE Transactions on Information Theory* 57.8, pp. 5455–5466 (cit. on p. 44).

Nielsen, Frank and Kazuki Okamura (2024). "Information measures and geometry of the hyperbolic exponential families of Poincaré and hyperboloid distributions". In: *Information Geometry* 7.Suppl 2, pp. 943–989 (cit. on p. 53).

Nijboer, Femke, Adrian Furdea, Ingo Gunst, Jürgen Mellinger, Dennis J McFarland, Niels Birbaumer, and Andrea Kübler (2008). "An auditory brain–computer interface (BCI)". In: *Journal of neuroscience methods* 167.1, pp. 43–50 (cit. on p. 26).

Norcia, Anthony M, L Gregory Appelbaum, Justin M Ales, Benoit R Cottereau, and Bruno Rossion (2015). "The steady-state visual evoked potential in vision research: A review". In: *Journal of vision* 15.6, pp. 4–4 (cit. on p. 19).

Nunez, Paul L and Ramesh Srinivasan (2006). *Electric fields of the brain: the neurophysics of EEG*. Oxford university press (cit. on p. 18).

Obermaier, Bernhard, Christoph Guger, Christa Neuper, and Gert Pfurtscheller (2001). "Hidden Markov models for online classification of single trial EEG data". In: *Pattern recognition letters* 22.12, pp. 1299–1309 (cit. on pp. 22, 23).

Obermaier, Bernhard, Christa Neuper, Christoph Guger, and Gert Pfurtscheller (2001). "Information transfer rate in a five-classes brain-computer interface". In: *IEEE Transactions on neural systems and rehabilitation engineering* 9.3, pp. 283–288 (cit. on pp. 22, 23).

Oikonomou, Vangelis P., Georgios Liaros, Kostantinos Georgiadis, Elisavet Chatzilari, Katerina Adam, Spiros Nikolopoulos, and Ioannis Kompatsiaris (2016). *Comparative evaluation of state-of-the-art algorithms for SSVEP-based BCIs* (cit. on p. 18).

Oswald, Antonin and Estelle Massart (2026). "Expressivity of congruence-based architectures for DNNs on positive-definite matrices". In: *arXiv preprint arXiv:2606.02490* (cit. on p. 57).

Pal, Subhadip, Subhajit Sengupta, Riten Mitra, and Arunava Banerjee (Sept. 2020). "Conjugate Priors and Posterior Inference for the Matrix Langevin Distribution on the Stiefel Manifold". In: *Bayesian Analysis* 15. DOI: 10.1214/19-BA1176 (cit. on p. 53).

Palaniappan, Ramaswamy (2005). "Brain-Computer interface design using band powers extracted during mental tasks". In: *Conference Proceedings. 2nd International IEEE EMBS Conference on Neural Engineering, 2005.* IEEE, pp. 321–324 (cit. on pp. 22, 23).

Pan, Yue-Ting, Jing-Lun Chou, and Chun-Shu Wei (2022). "MAtt: A manifold attention network for EEG decoding". In: *Advances in Neural Information Processing Systems* 35, pp. 31116–31129 (cit. on p. 57).

Pandit, J.J., J. Andrade, D.G. Bogod, J.M. Hitchman, W.R. Jonker, N. Lucas, J.H. Mackay, A.F. Nimmo, K. O'Connor, E.P. O'Sullivan, R.G. Paul, J.H.M.G. Palmer, F. Plaat, J.J. Radcliffe, M.R.J. Sury, H.E. Torevell, M. Wang, J. Hainsworth, T.M. Cook, James Armstrong, Jonathan Bird, Alison Eddy, William Harrop-Griffiths, Nicholas Love, Ravi Mahajan,

Abhiram Mallick, Ian Barker, Anahita Kirkpatrick, Jayne Molodynski, Karthikeyen Poonnusamy, Ramani Moonesinghe, David Weatherill, David Smith, Maddy Humphrey, Mary Casserly, Sharon Drake, and Jairaj Rangasami (2014). "5th National Audit Project (NAP5) on accidental awareness during general anaesthesia: summary of main findings and risk factors". In: *British Journal of Anaesthesia* 113.4, pp. 549–559. DOI: 10.1093/bja/aeu313 (cit. on pp. 24, 135).

Pandit, JJ and TM Cook (2013). "I. National Institute for Clinical Excellence guidance on measuring depth of anaesthesia: limitations of EEG-based technology". In: *British journal of anaesthesia* 110.3, pp. 325–328 (cit. on pp. 136, 254).

Papillon, Mathilde, Sophia Sanborn, Johan Mathe, Louisa Cornelis, Abby Bertics, Domas Buracas, Hansen J Lillemark, Christian Shewmake, Fatih Dinc, Xavier Pennec, et al. (2025). "Beyond euclid: An illustrated guide to modern machine learning with geometric, topological, and algebraic structures". In: *Machine Learning: Science and Technology* 6.3, p. 031002 (cit. on pp. 3, 46, 243).

Pawar, Niti and Odmara L Barreto Chang (2022). "Burst suppression during general anesthesia and postoperative outcomes: mini review". In: *Frontiers in systems neuroscience* 15, p. 767489 (cit. on p. 143).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830 (cit. on pp. 12, 103, 158).

Peng, Wei, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao (2021). "Hyperbolic deep neural networks: A survey". In: *IEEE Transactions on pattern analysis and machine intelligence* 44.12, pp. 10023–10044 (cit. on p. 58).

Peng, Zhen, Hongyi Li, Di Zhao, and Chengwei Pan (Jan. 2023). "Reducing the Dimensionality of SPD Matrices with Neural Networks in BCI". In: *Mathematics* 11.7, p. 1570. DOI: 10.3390/math11071570 (cit. on p. 50).

Pennec, Xavier (July 2006). "Intrinsic Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements". In: *Journal of Mathematical Imaging and Vision* 25.1, pp. 127–154. DOI: 10.1007/s10851-006-6228-4 (cit. on pp. 50, 51, 54, 56, 58, 64, 69, 73, 117, 246).

— (2015a). "Barycentric Subspaces Analysis on Spheres". In: *Mathematical Foundations of Computational Anatomy (MFCA'15)*, pp. 71–82 (cit. on p. 49).

— (Oct. 2015b). "Barycentric Subspaces and Affine Spans in Manifolds". In: *Lecture Notes in Computer Science*. Vol. 9389. Lecture Notes in Computer Science. Palaiseau, France, pp. 12–21. DOI: 10.1007/978-3-319-25040-3_2 (cit. on p. 49).

— (2019). "Curvature effects on the empirical mean in Riemannian and affine Manifolds: a non-asymptotic high concentration expansion in the small-sample regime". In: *arXiv preprint arXiv:1906.07418* (cit. on p. 72).

— (Jan. 1, 2020). "3 - Manifold-valued Image Processing with SPD Matrices". In: *Riemannian Geometric Statistics in Medical Image Analysis*. Ed. by Xavier Pennec, Stefan Sommer, and Tom Fletcher. Academic Press, pp. 75–134. DOI: 10.1016/B978-0-12-814725-2.00010-8 (cit. on pp. 4, 29–31, 34, 36–39, 46, 244, 245, 259).

Pereira, Michael, Ferran Argelaguet, José del R Millán, and Anatole Lécuyer (2018). "Novice shooters with lower pre-shooting alpha power have better performance during competition in a virtual reality scenario". In: *Frontiers in psychology* 9, p. 527 (cit. on p. 24).

Perera, Pramuditha, Poojan Oza, and Vishal M Patel (2021). "One-class classification: A survey". In: *arXiv preprint arXiv:2101.03064* (cit. on pp. 134, 255).

Perslev, Mathias, Sune Darkner, Lykke Kempfner, Miki Nikolic, Poul Jørgen Jennum, and Christian Igel (2021). "U-Sleep: resilient high-frequency sleep staging". In: *NPJ digital medicine* 4.1, p. 72 (cit. on pp. 22, 23).

Pewsey, Arthur and Eduardo García-Portugués (2021). "Recent advances in directional statistics". In: *Test* 30.1, pp. 1–58 (cit. on p. 52).

Peyré, Gabriel (2026). "Optimal and diffusion transports in machine learning". In: *International Congress of Mathematicians 2026*. SIAM, pp. 110–128 (cit. on pp. 128, 187).

Peyré, Gabriel and Marco Cuturi (2019). *Computational Optimal Transport: With Applications to Data Science*. Foundations and trends in machine learning. Now Publishers (cit. on pp. 41, 123).

Pfurtscheller, G. and F.H. Lopes da Silva (1999). "Event-related EEG/MEG synchronization and desynchronization: basic principles". In: *Clinical Neurophysiology* 110.11, pp. 1842–1857. DOI: 10.1016/S1388-2457(99)00141-8 (cit. on pp. 19, 22, 23, 28, 164, 238, 260).

Pfurtscheller, G. and C. Neuper (2001). "Motor imagery and direct brain-computer communication". In: *Proceedings of the IEEE* 89.7, pp. 1123–1134. DOI: 10.1109/5.939829 (cit. on pp. 19, 101, 185).

Pfurtscheller, Gert, Christa Neuper, GR Muller, Bernhard Obermaier, Gunter Krausz, A Schlogl, Reinhold Scherer, Bernhard Graimann, Claudia Keinrath, Dimitris Skliris, et al. (2003). "Graz-BCI: state of the art and clinical applications". In: *IEEE Transactions on neural systems and rehabilitation engineering* 11.2, pp. 1–4 (cit. on p. 16).

Poličar, Pavlin G., Martin Stražar, and Blaž Zupan (2024). "openTSNE: A Modular Python Library for t-SNE Dimensionality Reduction and Embedding". In: *Journal of Statistical Software* 109.3, pp. 1–30. DOI: 10.18637/jss.v109.i03 (cit. on p. 157).

Polyak, Adam, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. (2024). "Movie gen: A cast of media foundation models". In: *arXiv preprint arXiv:2410.13720* (cit. on pp. 112, 252).

Polyak, Boris T and Anatoli B Juditsky (1992). "Acceleration of stochastic approximation by averaging". In: *SIAM journal on control and optimization* 30.4, pp. 838–855 (cit. on p. 237).

Pooladian, Aram-Alexandre, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen (2023). "Multisample Flow Matching: Straightening Flows with Minibatch Couplings". In: *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 28100–28127 (cit. on p. 128).

Popivanov, D and A Mineva (1999). "Testing procedures for non-stationarity and non-linearity in physiological signals". In: *Mathematical biosciences* 157.1-2, pp. 303–320 (cit. on p. 25).

Punjasawadwong, Yodying, Aram Phongchiewboon, and Nutchanart Bunchungmongkol (2014). "Bispectral index for improving anaesthetic delivery and postoperative recovery". In: *Cochrane database of systematic reviews* 6 (cit. on p. 136).

Purdon, Patrick L, Aaron Sampson, Kara J Pavone, and Emery N Brown (2015). "Clinical electroencephalography for anesthesiologists part I: background and basic signatures". In: *Anesthesiology* 123.4, p. 937 (cit. on pp. 136, 140, 143).

Qin, Ruihan, Zhenxi Song, Huixia Ren, Zian Pei, Lin Zhu, Xue Shi, Yi Guo, Honghai Liu, Min Zhang, and Zhiguo Zhang (2024). "BNMTrans: A Brain Network Sequence-Driven Manifold-Based Transformer for Cognitive Impairment Detection Using EEG". In: *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2016–2020. DOI: 10.1109/ICASSP48485.2024.10447106 (cit. on p. 57).

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. (2021). "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PmLR, pp. 8748–8763 (cit. on p. 1).

Rakotomamonjy, Alain, Vincent Guigue, Gregory Mallet, and Victor Alvarado (2005). "Ensemble of SVMs for improving Brain-Computer interface P300 speller performances". In: *International conference on artificial neural networks*. Springer, pp. 45–50 (cit. on pp. 22, 23).

Ramos-Murguialday, Ander, Doris Broetz, Massimiliano Rea, Leonhard Läer, Özge Yilmaz, Fabricio L Brasil, Giulia Liberati, Marco R Curado, Eliana Garcia-Cossio, Alexandros Vyziotis, et al. (2013). "Brain–machine interface in chronic stroke rehabilitation: a controlled study". In: *Annals of neurology* 74.1, pp. 100–108 (cit. on p. 24).

Ramoser, Herbert, Johannes Muller-Gerking, and Gert Pfurtscheller (2000). "Optimal spatial filtering of single trial EEG during imagined hand movement". In: *IEEE transactions on rehabilitation engineering* 8.4, pp. 441–446 (cit. on pp. 21, 150).

Rao, C.R. (2009). *Linear Statistical Inference and its Applications*. Wiley Series in Probability and Statistics. Wiley (cit. on p. 52).

Riemann, Bernhard (1867). "Über die Hypothesen, welche der Geometrie zu Grunde liegen". In: *Abhandlungen der Königlichen Gesellschaft der Wissenschaften zu Göttingen* 13. Habilitationsvortrag, prononcée le 10 juin 1854 à Göttingen; publiée à titre posthume, communiquée par R. Dedekind, pp. 133–150 (cit. on pp. 2, 243).

Rimbert, Sébastien, Julien Lelarge, Philippe Guerci, Seyed Javad Bidgoli, Claude Meistelman, Guy Cheron, Ana Maria Cebolla Alvarez, and Denis Schmartz (2023). "Detection of motor cerebral activity after median nerve stimulation during general anesthesia (stim-motana): Protocol for a prospective interventional study". In: *JMIR research protocols* 12.1, e43870 (cit. on p. 136).

Rimbert, Sébastien, Pierre Riff, Nathalie Gayraud, Denis Schmartz, and Laurent Bougrain (2019). "Median nerve stimulation based BCI: a new approach to detect intraoperative awareness during general anesthesia". In: *Frontiers in Neuroscience* 13, p. 622 (cit. on pp. 134, 136).

Robert, Christian (1990). "Modified Bessel functions and their applications in probability and statistics". In: *Statistics and Probability Letters* 9.2, pp. 155–161. DOI: 10.1016/0167-7152(92)90011-S (cit. on p. 52).

Rotman, Joseph (1990). *Galois theory*. Springer (cit. on p. 156).

Rousseeuw, Peter J (1984). "Least median of squares regression". In: *Journal of the American statistical association* 79.388, pp. 871–880 (cit. on p. 29).

Roweis, Sam T. and Lawrence K. Saul (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding". In: *Science* 290.5500, pp. 2323–2326. DOI: 10.1126/science.290.5500.2323 (cit. on p. 46).

Roy, Raphaëlle N., Stéphane Bonnet, Sylvie Charbonnier, and Aurélie Campagne (July 2013). "Mental fatigue and working memory load estimation: Interaction and implications for EEG based passive BCI". In: *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference* 2013, pp. 6607–6610. DOI: 10.1109/EMBC.2013.6611070 (cit. on p. 26).

Roy, Raphaëlle N., Marcel F. Hinss, Ludovic Darmet, Simon Ladouce, Emilie S. Jahanpour, Bertille Somon, Xiaoqi Xu, Nicolas Drougard, Frederic Dehais, and Fabien Lotte (2022). "Retrospective on the First Passive Brain-Computer Interface Competition on Cross-Session Workload Estimation". In: *Frontiers in Neuroergonomics* 3. DOI: 10.3389/fnrgo.2022.838342 (cit. on pp. 26, 30).

Rozemberczki, Benedek, Lauren Watson, Péter Bayer, Hao Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar (2022). "The Shapley Value in Machine Learning". English. In: *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022*. Ed. by Luc {De Raedt} and Luc {De Raedt}. IJCAI International Joint Conference on Artificial Intelligence. Publisher Copyright: © 2022 International Joint Conferences

on Artificial Intelligence. All rights reserved.; 31st International Joint Conference on Artificial Intelligence, IJCAI 2022 ; Conference date: 23-07-2022 Through 29-07-2022. United States: International Joint Conferences on Artificial Intelligence, pp. 5572–5579. DOI: 10.24963/ijcai.2022/778 (cit. on p. 169).

Ruff, Lukas, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller (2021). "A unifying review of deep and shallow anomaly detection". In: *Proceedings of the IEEE* 109.5, pp. 756–795 (cit. on pp. 137, 145).

Rußwurm, Marc, Charlotte Pelletier, Maximilian Zollner, Sébastien Lefèvre, and Marco Körner (2020). "BreizhCrops: A Time Series Dataset for Crop Type Mapping". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences ISPRS (2020)* (cit. on pp. 102, 125, 238).

Sabbagh, David, Pierre Ablin, Gaël Varoquaux, Alexandre Gramfort, and Denis Engemann (2019). "Manifold-regression to predict from MEG/EEG brain signals without source modeling". In: *Advances in neural information processing systems* 32 (cit. on pp. 47, 58, 59, 179).

Saha, Simanto and Mathias Baumert (2020). "Intra-and inter-subject variability in EEG-based sensorimotor Brain-Computer interface: a review". In: *Frontiers in computational neuroscience* 13, p. 87 (cit. on pp. 26, 161).

Said, Salem, Lionel Bombrun, Yannick Berthoumieu, and Jonathan H Manton (2017). "Riemannian Gaussian distributions on the space of Symmetric Positive Definite matrices". In: *IEEE Transactions on Information Theory* 63.4, pp. 2153–2170 (cit. on pp. 8, 51, 54, 64, 69, 94, 95, 98, 113, 117, 137, 152, 176, 178, 180, 187, 246, 250, 258).

Said, Salem, Hatem Hajri, Lionel Bombrun, and Baba C. Vemuri (Feb. 2018). "Gaussian Distributions on Riemannian Symmetric Spaces: Statistical Learning With Structured Covariance Matrices". In: *IEEE Transactions on Information Theory* 64.2, pp. 752–772. DOI: 10.1109/TIT.2017.2713829 (cit. on pp. 50, 51, 55, 58).

Sakai, T. (1996). *Riemannian Geometry*. Fields Institute Communications. American Mathematical Soc. (cit. on pp. 89, 249).

Sannelli, Claudia, Carmen Vidaurre, Klaus-Robert Müller, and Benjamin Blankertz (Jan. 2019). "A large scale screening study with a SMR-based BCI: Categorization of BCI users and differences in their SMR activity". In: *PLOS ONE* 14.1, pp. 1–37. DOI: 10.1371/journal.pone.0207351 (cit. on p. 26).

Santos, Lucas Heck dos, João Guilherme Prado Barbon, Sylvain Chevallier, and Denis G Fantinato (Apr. 2026). "Non-Linear Activation Functions for Deep Riemannian Neural Networks". In: *Proc. ESANN'26*. Bruges, Belgium: Ciaco - i6doc.com, pp. 535–540. DOI: 10.14428/esann/2026.ES2026-312 (cit. on p. 56).

Saul, Lawrence K. and Sam T. Roweis (2003). "Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds". In: *Journal of Machine Learning Research* 4 (Jun), pp. 119–155 (cit. on p. 49).

Scherer, Reinhold, GR Muller, Christa Neuper, Bernhard Graimann, and Gert Pfurtscheller (2004). "An asynchronously controlled EEG-based virtual keyboard: improvement of the spelling rate". In: *IEEE Transactions on Biomedical Engineering* 51.6, pp. 979–984 (cit. on p. 22).

Schirrmeister, Robin Tibor, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball (2017). "Deep learning with convolutional neural networks for EEG decoding and visualization". In: *Human brain mapping* 38.11, pp. 5391–5420 (cit. on pp. 22, 23).

Schlögl, Alois, Carmen Vidaurre, and Klaus-Robert Müller (2010). "Adaptive methods in BCI research-an introductory tutorial". In: *Brain-Computer Interfaces: Revolutionizing*

Human-Computer Interaction. Ed. by Bernhard Graimann, Brendan Z. Allison, and Gert Pfurtscheller. The Frontiers Collection. Berlin, Heidelberg: Springer, pp. 331–355. DOI: 10.1007/978-3-642-02091-9_18 (cit. on pp. 22, 23).

Schnitzler, Alfons, Stephan Salenius, Riitta Salmelin, Veikko Jousmäki, and Riitta Hari (1997). "Involvement of primary motor cortex in motor imagery: a neuromagnetic study". In: *Neuroimage* 6.3, pp. 201–208 (cit. on p. 137).

Schwender, D, S Klasing, M Daunderer, C Madler, E Pöppel, and K Peter (1995). "Awareness during general anesthesia. Definition, incidence, clinical relevance, causes, avoidance and medicolegal aspects". In: *Der Anaesthesist* 44.11, pp. 743–754 (cit. on p. 135).

Seghier, Mohamed and Cathy Price (Mar. 2018). "Interpreting and Utilising Intersubject Variability in Brain Function". In: *Trends in Cognitive Sciences* 22. DOI: 10.1016/j.tics.2018.03.003 (cit. on p. 26).

Sei, Tomonari, Hiroki Shibata, Akimichi Takemura, Katsuyoshi Ohara, and Nobuki Takayama (2013). "Properties and applications of Fisher distribution on the rotation group". In: *Journal of Multivariate Analysis* 116, pp. 440–455 (cit. on p. 53).

Serre, D. (2002). *Matrices: Theory and Applications*. Graduate Texts in Mathematics. Springer (cit. on p. 121).

Shannon, Claude Elwood (1948). "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27, pp. 379–423 (cit. on p. 152).

Shapley, Lloyd S. (1953). "A value for n-person games". In: *Contributions to the Theory of Games, Volume II*. Ed. by Harold W. Kuhn and Albert W. Tucker. Annals of Mathematics Studies 28. Princeton, NJ: Princeton University Press, pp. 307–317 (cit. on p. 169).

Sharma, Vipul and Mitul Kumar Ahirwal (2024). "An end-to-end Brain-Computer interface system for mental workload estimation through hybrid deep learning model". In: *Human-Centric Intelligent Systems* 4.4, pp. 599–609 (cit. on p. 24).

Shenoy, Pradeep, Matthias Krauledat, Benjamin Blankertz, Rajesh PN Rao, and Klaus-Robert Müller (2006). "Towards adaptive classification for BCI". In: *Journal of neural engineering* 3.1, R13–R23 (cit. on pp. 22, 23).

Shiga, Kiyoshi (1984). "Hadamard Manifolds". In: *Geometry of Geodesics and Related Topics*. Ed. by Katsuhiro Shiohama. Vol. 3. Advanced Studies in Pure Mathematics. Nagara, Gifu, Japan: Mathematical Society of Japan, pp. 239–282. DOI: 10.2969/aspm/00310239 (cit. on pp. 87, 128).

Shimizu, Ryohei, Yusuke Mukuta, and Tatsuya Harada (2020). "Hyperbolic neural networks++". In: *arXiv preprint arXiv:2006.08210* (cit. on pp. 57, 58).

Siegel, Carl Ludwig (1943). "Symplectic Geometry". In: *American Journal of Mathematics* 65.1, pp. 1–86 (cit. on p. 34).

Silvoni, Stefano, Ander Ramos-Murguialday, Marianna Cavinato, Chiara Volpato, Giulia Cisotto, Andrea Turolla, Francesco Piccione, and Niels Birbaumer (2011). "Brain-computer interface in stroke: a review of progress". In: *Clinical EEG and neuroscience* 42.4, pp. 245–252 (cit. on p. 24).

Skovgaard, Lene Theil (1984). "A Riemannian Geometry of the Multivariate Normal Model". In: *Scandinavian Journal of Statistics* 11.4, pp. 211–223 (cit. on pp. 34, 36).

Skrodzki, Martin, Hunter van Geffen, Nicolas F Chaves-de-Plaza, Thomas Höllt, Elmar Eisemann, and Klaus Hildebrandt (2024). "Accelerating hyperbolic t-SNE". In: *IEEE transactions on visualization and computer graphics* 30.7, pp. 4403–4415 (cit. on p. 50).

Slutzky, Marc, Mariska Vansteensel, Christian Herff, and Robert Gaunt (June 2025). "A brain–computer interface working definition". In: *Nature Biomedical Engineering* 9. DOI: 10.1038/s41551-025-01414-8 (cit. on p. 16).

Smith, Alexander, Jinxi Hua, Benjamin de Foy, James Schauer, and Victor Zavala (Dec. 2022). *Multi-Site, Multi-Pollutant Atmospheric Data Analysis Using Riemannian Geometry*. SSRN Scholarly Paper. Rochester, NY. DOI: 10.2139/ssrn.4309521 (cit. on pp. 102, 108).

Smith, Alexander, Benjamin Laubach, Ivan Castillo, and Victor M. Zavala (2022). "Data analysis using Riemannian geometry and applications to chemical engineering". In: *Computers & Chemical Engineering* 168, p. 108023. DOI: https://doi.org/10.1016/j.compchemeng.2022.108023 (cit. on pp. 102, 238).

Snedecor, George W. (1934). *Calculation and Interpretation of Analysis of Variance and Covariance*. Ames, Iowa: Collegiate Press (cit. on p. 184).

Sommer, Stefan, François Lauze, Søren Hauberg, and Mads Nielsen (2010). "Manifold valued statistics, exact principal, geodesic analysis and the effect of linear, approximations". In: *Proceedings of the 11th European Conference on Computer Vision: Part VI*. ECCV'10. Heraklion, Crete, Greece: Springer-Verlag, pp. 43–56 (cit. on p. 49).

Sommer, Stefan, François Lauze, and Mads Nielsen (2014). "Optimization over geodesics for exact principal geodesic analysis". In: *Advances in Computational Mathematics* 40.2, pp. 283–313 (cit. on pp. 49, 58, 179).

Song, Yang, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole (2021). "Score-Based Generative Modeling through Stochastic Differential Equations". In: *International Conference on Learning Representations* (cit. on pp. 1, 112, 251).

Song, Yonghao, Qingqing Zheng, Bingchuan Liu, and Xiaorong Gao (2022). "EEG conformer: Convolutional transformer for EEG decoding and visualization". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 31, pp. 710–719 (cit. on pp. 22, 23).

Spinner, Jonas, Victor Bresó, Pim De Haan, Tilman Plehn, Jesse Thaler, and Johann Brehmer (2024). "Lorentz-equivariant geometric algebra transformers for high-energy physics". In: *Advances in neural information processing systems* 37, pp. 22178–22205 (cit. on p. 5).

Sra, Suvrit (2016a). "Directional statistics in machine learning: a brief review". In: *arXiv preprint arXiv:1605.00316* (cit. on p. 52).

— (2016b). "Positive definite matrices and the S-divergence". In: *Proceedings of the American Mathematical Society* 144.7, pp. 2787–2797 (cit. on pp. 44, 48).

Sra, Suvrit and Reshad Hosseini (2015). "Conic geometric optimization on the manifold of positive definite matrices". In: *SIAM Journal on Optimization* 25.1, pp. 713–739 (cit. on pp. 37, 87).

Stam, Cornelis J (2005). "Nonlinear dynamical analysis of EEG and MEG: review of an emerging field". In: *Clinical neurophysiology* 116.10, pp. 2266–2301 (cit. on p. 25).

Steyrl, David, Reinhold Scherer, Josef Faller, and Gernot R. Müller-Putz (2016). "Random forests in non-invasive sensorimotor rhythm brain-computer interfaces: a practical and convenient non-linear classifier". In: *Biomedical Engineering / Biomedizinische Technik* 61.1, pp. 77–86. DOI: doi:10.1515/bmt-2014-0117 (cit. on pp. 125, 238).

Stokes, Jonathan M, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. (2020). "A deep learning approach to antibiotic discovery". In: *Cell* 180.4, pp. 688–702 (cit. on p. 3).

Stretanski, Michael F., Alexander M. Dydyk, and Marco Cascella (Jan. 2026). *Median Nerve Injury*. In: StatPearls [Internet]. Updated 2025 Jul 7. Treasure Island (FL) (cit. on p. 136).

Štrumbelj, Erik and Igor Kononenko (2011). "A general method for visualizing and explaining black-box regression models". In: *International conference on adaptive and natural computing algorithms*. Springer, pp. 21–30 (cit. on p. 169).

— (2014). "Explaining prediction models and individual predictions with feature contributions". In: *Knowledge and information systems* 41.3, pp. 647–665 (cit. on p. 169).

Surrel, Thibault de, Aimi Okabayashi, and Nicolas Courty (2026). "Wrapped Flow Matching for Symmetric Positive Definite Matrices". In: *Transactions on Machine Learning Research* (cit. on pp. 9, 113).

Sylvester, James Joseph (1884). “Sur l’équation en matrices $px = xq$”. French. In: *Comptes Rendus de l’Académie des Sciences* 99.2, pp. 67–71, 115–116 (cit. on p. 41).

Symeonidou, Evangelia-Regkina, Andrew D Nordin, W David Hairston, and Daniel P Ferris (2018). “Effects of cable sway, electrode surface area, and electrode mass on electroencephalography signal quality during motion”. In: *Sensors* 18.4, p. 1073 (cit. on p. 18).

Takatsu, Asuka (2011). “Wasserstein geometry of Gaussian measures”. In: *Osaka Journal of Mathematics* 48.4, pp. 1005–1026 (cit. on p. 41).

Talukdar, Upasana, Shyamanta M. Hazarika, and John Q. Gan (2020). “Adaptation of Common Spatial Patterns based on mental fatigue for motor-imagery BCI”. In: *Biomedical Signal Processing and Control* 58, p. 101829. DOI: 10.1016/j.bspc.2019.101829 (cit. on p. 26).

Taminiau, Timothé, Estelle Massart, and Geovani Nunes Grapiglia (2026). “Riemannian optimization with finite-difference gradient approximations”. In: *arXiv preprint arXiv:2601.08751* (cit. on p. 171).

Tanaka, Toshihisa, Takashi Uehara, and Yuichi Tanaka (June 2016). “Dimensionality Reduction of Sample Covariance Matrices by Graph Fourier Transform for Motor Imagery Brain-Machine Interface”. In: *2016 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 1–5. DOI: 10.1109/SSP.2016.7551798 (cit. on p. 50).

Tangermann, Michael, Sylvain Chevallier, Matthias Dold, Pierre Guetschel, Reinmar Kobler, Theodore Papadopoulo, and Jordy Thielen (2025). “Learning from small datasets—review of workshop 6 of the 10th International BCI Meeting 2023”. In: *Journal of Neural Engineering* 22.3, p. 033001. DOI: 10.1088/1741-2552/addf80 (cit. on p. 27).

Tangermann, Michael, Klaus-Robert Müller, Ad Aertsen, Niels Birbaumer, Christoph Braun, Clemens Brunner, Robert Leeb, Carsten Mehring, Kai J. Miller, Gernot Mueller-Putz, Guido Nolte, Gert Pfurtscheller, Hubert Preissl, Gerwin Schalk, Alois Schlögl, Carmen Vidaurre, Stephan Waldert, and Benjamin Blankertz (2012). “Review of the BCI Competition IV”. In: *Frontiers in Neuroscience* 6. DOI: 10.3389/fnins.2012.00055 (cit. on pp. 18, 20, 25, 108, 160, 161, 164–166, 170, 171, 239, 240, 260).

Tasbihgou, SR, MF Vogels, and AR Absalom (2018). “Accidental awareness during general anaesthesia–a narrative review”. In: *Anaesthesia* 73.1, pp. 112–122 (cit. on pp. 136, 254).

Tenenbaum, Joshua B., Vin de Silva, and John C. Langford (2000). “A Global Geometric Framework for Nonlinear Dimensionality Reduction”. In: *Science* 290.5500, p. 2319 (cit. on p. 46).

Teplan, Michal et al. (2002). “Fundamentals of EEG measurement”. In: *Measurement science review* 2.2, pp. 1–11 (cit. on p. 18).

Terras, Audrey (1988). *Harmonic Analysis on Symmetric Spaces and Applications II*. New York, NY: Springer. DOI: 10.1007/978-1-4612-3820-1 (cit. on pp. i, 37, 227).

Thanwerdas, Yann and Xavier Pennec (2023a). “Bures–Wasserstein minimizing geodesics between covariance matrices of different ranks”. In: *SIAM Journal on Matrix Analysis and Applications* 44.3, pp. 1447–1476 (cit. on p. 41).

— (2023b). “O(n)-invariant Riemannian metrics on SPD matrices”. In: *Linear Algebra and its Applications* 661, pp. 163–201. DOI: 10.1016/j.laa.2022.12.009 (cit. on p. 44).

Thomas, David B, Wayne Luk, Philip HW Leong, and John D Villasenor (2007). “Gaussian random number generators”. In: *ACM Computing Surveys (CSUR)* 39.4, 11–es (cit. on p. 112).

Thompson, Margaret C (2019). “Critiquing the Concept of BCI Illiteracy: MC Thompson”. In: *Science and engineering ethics* 25.4, pp. 1217–1233 (cit. on p. 26).

Timoz, Dimitri, **Thibault de Surrel**, and Florian Yger (2026). “Riemannian adversarial attacks on Symmetric Positive Definite matrices”. In: *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 5356–5360 (cit. on pp. 11, 170, 171).

Tipping, Michael E and Christopher M Bishop (1999). "Probabilistic principal component analysis". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 61.3, pp. 611–622 (cit. on p. 186).

Tong, Alexander, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio (2024). "Improving and generalizing flow-based generative models with minibatch optimal transport". In: *Transactions on Machine Learning Research*. Expert Certification (cit. on pp. 117, 119).

Tong, Alexander Y., Nikolay Malkin, Kilian Fatras, Lazar Atanackovic, Yanlei Zhang, Guillaume Huguet, Guy Wolf, and Yoshua Bengio (2024). "Simulation-Free Schrödinger Bridges via Score and Flow Matching". In: *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*. Ed. by Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li. Vol. 238. Proceedings of Machine Learning Research. PMLR, pp. 1279–1287 (cit. on p. 128).

Townsend, James, Niklas Koep, and Sebastian Weichwald (2016). "Pymanopt: A Python Toolbox for Optimization on Manifolds using Automatic Differentiation". In: *Journal of Machine Learning Research* 17.137, pp. 1–5 (cit. on p. 85).

Tran, Ly V, Hieu M Tran, Tuan M Le, Tri TM Huynh, Hung T Tran, and Son VT Dao (2022). "Application of machine learning in epileptic seizure detection". In: *Diagnostics* 12.11, p. 2879 (cit. on p. 134).

Trocellier, David, Juliette Meunier, Loïc Bechon, Philomène Motti, Bernard N'Kaoua, and Fabien Lotte (May 2025). *BRAINKART : A tangible open-source fast calibration BCI for science popularisation*. CORTICO 2025 - Journées pour le COllectif pour la Recherche Transdisciplinaire sur les Interfaces Cerveau-Ordinateur. Poster (cit. on p. 24).

Trocellier, David, Bernard N'Kaoua, and Fabien Lotte (2024). "Visual cues can bias EEG deep learning models". In: *Neuroergonomics 2024-The 5th International Conference* (cit. on pp. 148, 258).

Troshin, Sergey and Vlad Niculae (2023). "Wrapped beta-Gaussians with compact support for exact probabilistic modeling on manifolds". In: *Transactions on Machine Learning Research* (cit. on pp. 55, 67, 80, 175, 248).

Tuzel, Oncel, Fatih Porikli, and Peter Meer (2006). "Region covariance: A fast descriptor for detection and classification". In: *European conference on computer vision*. Springer, pp. 589–600 (cit. on p. 4).

— (2008). "Pedestrian Detection via Classification on Riemannian Manifolds". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10, pp. 1713–1727. DOI: 10.1109/TPAMI.2008.75 (cit. on p. 46).

Tyler, David E (1987). "A distribution-free M-estimator of multivariate scatter". In: *The annals of Statistics*, pp. 234–251 (cit. on p. 29).

Unwin, Antony (Jan. 2020). "Why Is Data Visualization Important? What Is Important in Data Visualization?" In: *Harvard Data Science Review* 2.1. https://hdsr.mitpress.mit.edu/pub/zok97i7p (cit. on p. 149).

Uusitalo, Mikko A and Risto J Ilmoniemi (1997). "Signal-space projection method for separating MEG or EEG into components". In: *Medical and biological engineering and computing* 35.2, pp. 135–140 (cit. on pp. 19, 104).

Van Assel, Hugues, Titouan Vayer, Rémi Flamary, and Nicolas Courty (May 2023). *SNEkhorn: Dimension Reduction with Symmetric Entropic Affinities* (cit. on p. 239).

Vanacker, Gerolf, José del R Millán, Eileen Lew, Pierre W Ferrez, Ferran Galán Moles, Johan Philips, Hendrik Van Brussel, and Marnix Nuttin (2007). "Context-Based Filtering for Assisted Brain-Actuated Wheelchair Driving". In: *Computational intelligence and neuroscience* 2007.1, p. 025130 (cit. on p. 23).

Vapnik, Vladimir (2013). *The nature of statistical learning theory*. Springer science & business media (cit. on pp. 6, 244).

Värbu, Kaido, Naveed Muhammad, and Yar Muhammad (2022). "Past, present, and future of EEG-based BCI applications". In: *Sensors* 22.9, p. 3331 (cit. on p. 16).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30 (cit. on p. 57).

Velut, Sébastien, Jordy Thielen, Sylvain Chevallier, Marie-Constance Corsi, and Frédéric Dehais (2026). "Neurophysiological screening of individual variability for robust decoding in c-VEP-based BCI". In: *Imaging Neuroscience* 4, IMAG–a (cit. on p. 19).

Venna, Jarkko and Samuel Kaski (2001). "Neighborhood preservation in nonlinear projection methods: An experimental study". In: *International conference on artificial neural networks*. Springer, pp. 485–491 (cit. on pp. 159, 259).

Villani, Cédric (2003). *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society (cit. on p. 41).

— (2009). *Optimal transport: old and new*. Vol. 338. Springer (cit. on p. 187).

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors (2020). "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 38).

Vyas, Apoorv, Bowen Shi, Matthew Le, Andros Tjandra, Yi-Chiao Wu, Baishan Guo, Jiemin Zhang, Xinyue Zhang, Robert Adkins, William Ngan, et al. (2023). "Audiobox: Unified audio generation with natural language prompts". In: *arXiv preprint arXiv:2312.15821* (cit. on pp. 112, 252).

Wagner vom Berg, Gabriel (2025). "Kernels on the Riemannian Manifold of Symmetric Positive Definite Matrices". PhD thesis. Technische Universität Berlin (cit. on pp. 48, 106, 108, 251).

Wagner vom Berg, Gabriel, Vera Röhr, Daniel Platt, and Benjamin Blankertz (2025). "A New Canonical Log-Euclidean Kernel for Symmetric Positive Definite Matrices for EEG Analysis (Oct 2024)". In: *IEEE Transactions on Biomedical Engineering* 72.3, pp. 1000–1007. DOI: 10.1109/TBME.2024.3483936 (cit. on pp. 48, 58, 108).

Wang, Rui, Xiao-Jun Wu, Ziheng Chen, Tianyang Xu, and Josef Kittler (Dec. 2022). "DreamNet: A Deep Riemannian Manifold Network for SPD Matrix Learning". In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pp. 3241–3257 (cit. on pp. 57, 159).

Wang, Xiuheng, Ricardo Borsoi, Arnaud Breloy, and Cédric Richard (2025). "Riemannian change point detection on manifolds with robust centroid estimation". In: *2025 33rd European Signal Processing Conference (EUSIPCO)*. IEEE, pp. 1–5 (cit. on pp. 106, 145).

Wang, Xiuheng, Ricardo Augusto Borsoi, and Cédric Richard (2024). "Non-parametric online change point detection on Riemannian manifolds". In: *Forty-first International Conference on Machine Learning* (cit. on pp. 105, 143, 145).

Wang, Zhizhou, B.C. Vemuri, Y. Chen, and T.H. Mareci (2004). "A constrained variational principle for direct estimation and smoothing of the diffusion tensor field from complex DWI". In: *IEEE Transactions on Medical Imaging* 23.8, pp. 930–939. DOI: 10.1109/TMI.2004.831218 (cit. on pp. 40, 245).

Wasserman, Larry (2004). *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics. New York: Springer. DOI: 10.1007/978-0-387-21736-9 (cit. on pp. 28, 63, 69, 74, 75, 82, 85, 246, 249).

Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). "How to Use t-SNE Effectively". In: *Distill*. DOI: 10.23915/distill.00002 (cit. on pp. 151, 163, 239).

Wishart, John (1928). "The Generalised Product Moment Distribution in Samples from a Normal Multivariate Population". In: *Biometrika* 20.1/2. Part A, pp. 32–52 (cit. on pp. 53, 55, 58, 187).

Wolpaw, Jonathan R, Dennis J McFarland, Gregory W Neat, and Catherine A Forneris (1991). "An EEG-based brain-computer interface for cursor control". In: *Electroencephalography and clinical neurophysiology* 78.3, pp. 252–259 (cit. on p. 23).

Woo, Jeheon, Seonghwan Kim, Jun Hyeong Kim, and Woo Youn Kim (2026). "Riemannian denoising model for molecular structure optimization with chemical accuracy". In: *Nature Computational Science* 6.2, pp. 134–144 (cit. on p. 187).

Xie, Xiaofeng, Zhu Liang Yu, Haiping Lu, Zhenghui Gu, and Yuanqing Li (June 2017). "Motor Imagery Classification Based on Bilinear Sub-Manifold Learning of Symmetric Positive-Definite Matrices". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.6, pp. 504–516. DOI: 10.1109/TNSRE.2016.2587939 (cit. on p. 50).

Xu, Jonathan, Ugo Bruzadin Nunes, Wangshu Jiang, Samuel Ryther, Jordan Pringle, Paul S Scotti, Arnaud Delorme, and Reese Kneeland (2025). "Alljoined-1.6 M: A Million-Trial EEG-Image Dataset for Evaluating Affordable Brain-Computer Interfaces". In: *arXiv preprint arXiv:2508.18571* (cit. on p. 26).

Xu, L, A-S Wu, and Y Yue (2009). "The incidence of intra-operative awareness during general anesthesia in China: a multi-center observational study". In: *Acta Anaesthesiologica Scandinavica* 53.7, pp. 873–882 (cit. on p. 136).

Yger, Florian (2013). "A review of kernels on covariance matrices for BCI applications". In: *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. DOI: 10.1109/MLSP.2013.6661972 (cit. on pp. 47, 106).

Yger, Florian, Maxime Berar, and Fabien Lotte (2017). "Riemannian Approaches in Brain-Computer Interfaces: A Review". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25.10, pp. 1753–1762. DOI: 10.1109/TNSRE.2016.2627016 (cit. on pp. 4, 30, 34, 46, 244).

Yger, Florian, Fabien Lotte, and Masashi Sugiyama (2015). "Averaging covariance matrices for EEG signal classification based on the CSP: An empirical study". In: *2015 23rd European Signal Processing Conference (EUSIPCO)*, pp. 2721–2725. DOI: 10.1109/EUSIPCO.2015.7362879 (cit. on p. 21).

Yger, Florian and Masashi Sugiyama (2015). *Supervised LogEuclidean Metric Learning for Symmetric Positive Definite Matrices* (cit. on pp. 179, 180).

Ying, Rex, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec (2018). "Graph convolutional neural networks for web-scale recommender systems". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–983 (cit. on p. 3).

Zaghen, Olga, Floor Eijkelboom, Alison Pouplin, Cong Liu, Max Welling, Jan-Willem van de Meent, and Erik J. Bekkers (2025). *Riemannian Variational Flow Matching for Material and Protein Design* (cit. on p. 113).

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola (2017). "Deep sets". In: *Advances in neural information processing systems* 30 (cit. on p. 5).

Zhang, Miaomiao and Tom Fletcher (2013). "Probabilistic Principal Geodesic Analysis". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc. (cit. on p. 49).

Zheng, Jingyi, Huajun Huang, Yuyan Yi, Yuexin Li, and Shu-Chin Lin (2023). "Barycenter estimation of positive semi-definite matrices with Bures-Wasserstein distance". In: *arXiv preprint arXiv:2302.14618* (cit. on p. 41).

Zhou, Bangyan, Xiaopei Wu, Zhao Lv, Lei Zhang, and Xiaojin Guo (Sept. 2016). "A Fully Automated Trial Selection Method for Optimization of Motor Imagery Based Brain-Computer Interface". In: *PLOS ONE* 11.9, pp. 1–20. DOI: 10.1371/journal.pone.0162657 (cit. on pp. 101, 108).

Zhou, Jie, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun (2020). "Graph neural networks: A review of methods and applications". In: *AI open* 1, pp. 57–81 (cit. on p. 1).

Zhou, Yuansheng and Tatyana O Sharpee (2021). "Hyperbolic geometry of gene expression". In: *Iscience* 24.3 (cit. on pp. 50, 58).

Zhou, Zhixuan, Tingting Dan, and Guorong Wu (2025). "BrainFlow: A Holistic Pathway of Dynamic Neural System on Manifold". In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems* (cit. on pp. 113, 122).

Zhu, Danhua, Jordi Bieger, Gary Garcia Molina, and Ronald M Aarts (2010). "A survey of stimulation methods used in SSVEP-based BCIs". In: *Computational intelligence and neuroscience* 2010.1, p. 702357 (cit. on p. 19).

Appendices

Appendix A

Proofs

Contents

A.1 Proofs of Chapter 2 225
A.2 Proofs of Chapter 3 227
A.2.1 The closed-form of the Jacobian of the exponential map on $\mathcal{P}_d$ . . . 227
A.2.2 The identifiability of the wrapped Gaussian on $\mathbb{H}^d$ 229
A.3 Proofs of Chapter 5 230
A.3.1 The proof of the general conditional vector field 230
A.4 Proofs of Chapter 7 231
A.4.1 Proof of the Riemannian gradient of the t-SNE 231

In this first appendix, we provide the proofs of the unproved results in the main text. We start by recalling the different statements and then give the proofs.

A.1 Proofs of Chapter 2

Proposition 2.22. *The Riemannian volume form of $\mathcal{P}_d$ equipped with the AIRM metric is given by*

$$\mathrm{d}\,\mathrm{vol}(\boldsymbol{X}) = 2^{\frac{d(d-1)}{4}} \det(\boldsymbol{X})^{-\frac{d+1}{2}} \prod_{i \leqslant j} \mathrm{d}X_{ij},$$

where $\boldsymbol{X} = (X_{ij})_{1 \leqslant i,j \leqslant d} \in \mathcal{P}_d$ and $\mathrm{d}X_{ij}$ are the Lebesgue measures on $\mathbb{R}$.

Proof. Based on Proposition 2.41 of J. M. Lee (2018), in order to compute the Riemannian volume form and given oriented local coordinates $(\boldsymbol{X}^1, \cdots, \boldsymbol{X}^D)$ (as we denote $D = \dim \mathcal{P}_d = d(d+1)/2$), one has

$$\mathrm{d}\,\mathrm{vol}(\boldsymbol{X}) = \sqrt{\det(g_{ij})}\, \mathrm{d}\boldsymbol{X}^1 \wedge \cdots \wedge \mathrm{d}\boldsymbol{X}^D,$$

where (g_{ij}) are the local coordinates of the Riemannian metric characterized by $g_{ij}(\boldsymbol{X}) = \langle \partial_i, \partial_j \rangle_{\boldsymbol{X}}$ with ∂_i the i-th coordinate vector field. In the case of the manifold $\mathcal{P}_d$, we consider the basis $(\boldsymbol{U}_{ij})_{i \leqslant j}$ of the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$, where $\boldsymbol{U}_{ij}$ is the matrix whose entries are zero everywhere except: if $i \neq j$, it has a value 1 in positions (i, j) and (j, i); if $i = j$, it has a value 1 in position (i, i).

Let us first consider the case where the matrix $\boldsymbol{X}$ is diagonal: $\boldsymbol{X} = \operatorname{diag}(x_1, ..., x_d)$. After doing some painful computations, one can show that in this case, for $\boldsymbol{U}$ and $\boldsymbol{V}$ two tangent vectors at $\boldsymbol{X}$ (thus symmetric matrices), one has:

$$\langle \boldsymbol{U}, \boldsymbol{V} \rangle_{\boldsymbol{X}} = \sum_{l=1}^{d} \sum_{k=1}^{d} u_{lk} v_{kl} x_l^{-1} x_k^{-1}.$$

Thus, applying this formula to $\boldsymbol{U}_{ij}$ and $\boldsymbol{U}_{mn}$, one has

$$\langle \boldsymbol{U}_{ij}, \boldsymbol{U}_{mn} \rangle_{\boldsymbol{X}} = \sum_{l=1}^{d} \sum_{k=1}^{d} (\boldsymbol{U}_{ij})_{lk} (\boldsymbol{U}_{mn})_{kl} x_l^{-1} x_k^{-1}.$$

We need to distinguish two cases. If $i = j$, then $\boldsymbol{U}_{ii}$ has a single nonzero entry, located at position (i, i). Hence, the only term in the sum that may be nonzero is $(l, k) = (i, i)$, which requires $(\boldsymbol{U}_{mn})_{ii} = 1$ and therefore forces $m = n = i$. Thus,

$$\langle \boldsymbol{U}_{ii}, \boldsymbol{U}_{mm} \rangle_{\boldsymbol{X}} = \delta_{i,m}\, x_i^{-2}.$$

If $i < j$, then $\boldsymbol{U}_{ij}$ has two nonzero entries, located at positions (i, j) and (j, i). The sum is nonzero only when $(\boldsymbol{U}_{mn})_{kl} = 1$ at one of those positions, which forces $\{m, n\} = \{i, j\}$, i.e. (since $m \leqslant n$) $(m, n) = (i, j)$. The two positions $(l, k) = (i, j)$ and $(l, k) = (j, i)$ each contribute $x_i^{-1} x_j^{-1}$, so

$$\langle \boldsymbol{U}_{ij}, \boldsymbol{U}_{mn} \rangle_{\boldsymbol{X}} = 2\, \delta_{(i,j)=(m,n)}\, x_i^{-1} x_j^{-1} \qquad (i < j,\ m < n).$$

Finally, the cross terms ($i = j$ with $m < n$, or $i < j$ with $m = n$) vanish, since $\boldsymbol{U}_{ii}$ is supported on the diagonal whereas $\boldsymbol{U}_{mn}$ for $m < n$ is supported off-diagonal. Therefore, the Gram matrix (g_{ij}) is diagonal in the basis $(\boldsymbol{U}_{ij})_{i \leqslant j}$, with diagonal entries x_i^{-2} for $i = j$ and $2x_i^{-1} x_j^{-1}$ for $i < j$. Its determinant is given by

$$\det(g_{ij}) = 2^{\frac{d(d-1)}{2}} \prod_{i\ \leqslant j} x_i^{-1} x_j^{-1}.$$

Using the natural logarithm to simplify the computations, one has

$$\begin{aligned}
\log \prod_{i\ \leqslant j} x_i^{-1} x_j^{-1} &= -\sum_{i \leqslant j} (\log x_i + \log x_j) \\
&= -\sum_{i \leqslant j} \log x_i - \sum_{j \leqslant i} \log x_i \\
&= -\sum_{i,j} \log x_i - \sum_{i} \log x_i \\
&= -(d+1) \log \prod_{i} x_i = -(d+1) \log(\det \boldsymbol{X}).
\end{aligned}$$

Thus, $\sqrt{\det(g_{ij})} = 2^{\frac{d(d-1)}{4}} (\det \boldsymbol{X})^{-(d+1)/2}$. Identifying the volume form on the oriented chart $\boldsymbol{X} \mapsto (X_{ij})_{i \leqslant j}$ with the corresponding positive Borel measure (i.e., replacing the wedge $\mathrm{d}\boldsymbol{X}^1 \wedge \cdots \wedge \mathrm{d}\boldsymbol{X}^D$ with the product Lebesgue measure $\prod_{i \leqslant j} \mathrm{d}X_{ij}$), we obtain the stated formula.

For the general case of a full matrix $\boldsymbol{X}$, one has that as the AIRM metric is invariant under affine transformations, its associated volume form must also be invariant. Using the

spectral theorem given in Theorem 2.2, one can decompose the matrix $\boldsymbol{X}$ as $\boldsymbol{X} = \boldsymbol{U}\Lambda\boldsymbol{U}^\top$ where $\boldsymbol{U}$ is orthogonal and Λ is diagonal. Thus, using the diagonal case, one has that

$$\begin{aligned}\mathrm{d}\,\mathrm{vol}(\boldsymbol{X}) &= \mathrm{d}\,\mathrm{vol}(\Lambda) \\ &= 2^{\frac{d(d-1)}{4}} \det(\Lambda)^{-\frac{d+1}{2}} \prod_{i\leqslant j} \mathrm{d}X_{ij} = 2^{\frac{d(d-1)}{4}} \det(\boldsymbol{X})^{-\frac{d+1}{2}} \prod_{i\leqslant j} \mathrm{d}X_{ij},\end{aligned}$$

as det is invariant too. □

Remark A.1. A reader checking Proposition 2.22 against Equation (1.16) of Terras (1988, Section 4.1.3) will notice that the constant $2^{\frac{d(d-1)}{4}}$ is absent there. There is no contradiction: Terras *defines* the $\mathrm{GL}(d)$-invariant measure on $\mathcal{P}_d$ to be $\det(\boldsymbol{X})^{-\frac{d+1}{2}} \prod_{i\leqslant j} \mathrm{d}X_{ij}$, and such an invariant measure is only unique up to a positive multiplicative constant. The factor $2^{\frac{d(d-1)}{4}}$ is precisely what relates that normalization to the Riemannian volume form of the AIRM metric.

A.2 Proofs of Chapter 3

A.2.1 The closed-form of the Jacobian of the exponential map on $\mathcal{P}_d$

Proposition 3.4 (Jacobian of the exponential map)**.** *The Jacobian determinant of the exponential map* Exp_{I_d} *at the identity is:*

$$\forall \boldsymbol{U} \in T_{I_d}\mathcal{P}_d,\ J_{I_d}(\boldsymbol{U}) = 2^{\frac{d(d-1)}{2}} \prod_{i<j} \frac{\sinh\left(\frac{\lambda_i(\boldsymbol{U})-\lambda_j(\boldsymbol{U})}{2}\right)}{\lambda_i(\boldsymbol{U}) - \lambda_j(\boldsymbol{U})} \tag{3.3}$$

where the $\lambda_i(\boldsymbol{U})$ *are the eigenvalues of* $\boldsymbol{U}$*. Then, one can use the previous formula to compute the Jacobian determinant of the exponential map at any point* $\boldsymbol{P} \in \mathcal{P}_d$*:*

$$\forall \boldsymbol{U} \in T_{\boldsymbol{P}}\mathcal{P}_d,\ J_{\boldsymbol{P}}(\boldsymbol{U}) = J_{I_d}(\boldsymbol{P}^{-1/2}\boldsymbol{U}\boldsymbol{P}^{-1/2}).$$

Proof. We will start by proving the formula for the Jacobian determinant of the exponential map at the identity Exp_{I_d}. When the tangent plan of interest is at the identity I_d, the Riemannian exponential map Exp_{I_d} is simply the matrix exponential exp:

$$\mathrm{Exp}_{I_d} : \boldsymbol{U} \in T_{I_d}\mathcal{P}_d \mapsto \exp(\boldsymbol{U}) \in \mathcal{P}_d.$$

One can see this result using the expression of the Riemannian exponential given in Proposition 2.19. To prove the relation Equation (3.3), we will start by the case where $\boldsymbol{U}$ is a diagonal matrix. We will then extend the result to the general case.

Case 1: $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d$ diagonal Let us consider $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d$ diagonal, $\boldsymbol{U} = \mathrm{diag}(\lambda_1, ..., \lambda_d)$. In the following, we will denote by Ψ the differential of the Riemannian exponential in $\boldsymbol{U}$: $\Psi = \mathrm{d}\,\mathrm{Exp}_{I_d}(\boldsymbol{U}) = \mathrm{d}\exp(\boldsymbol{U})$. We therefore want to compute the determinant of Ψ: $\det\Psi$. One has that $\Psi : T_{I_d}\mathcal{P}_d \to T_{\exp(\boldsymbol{U})}\mathcal{P}_d$, where we have identified $T_{\boldsymbol{U}}T_{I_d}\mathcal{P}_d$ with $T_{I_d}\mathcal{P}_d$. To compute the determinant, one needs to choose adequate bases in both tangent spaces $T_{I_d}\mathcal{P}_d$ and $T_{\exp(\boldsymbol{U})}\mathcal{P}_d$. By "adequate", we mean that the transformation between the two bases does not imply any volume change. For this, we consider for $T_{I_d}\mathcal{P}_d$ the basis $(\boldsymbol{E}_{ij}^{I_d})_{i\leqslant j}$ and for $T_{\exp(\boldsymbol{U})}\mathcal{P}_d$ the basis $(\boldsymbol{E}_{ij}^{\exp(\boldsymbol{U})})_{i\leqslant j}$ as defined in Equation (2.6). Using the expression of the parallel transport given at Proposition 2.20, the transformation from the first to the second basis is the parallel transport, which does not imply any volume changes, as the parallel transport is a linear isometry (see Proposition 10.36 of Boumal 2023).

Now that we have our two basis, we want to compute the matrix of Ψ in those two bases. For this, we need to compute $\Psi(\boldsymbol{E}_{ij}^{I_d})$ and express it in the basis $(\boldsymbol{E}_{ij}^{\exp(\boldsymbol{U})})_{i\leqslant j}$. As $\boldsymbol{U}$ is diagonal, we can use the *Daletskii-Krein formula* (see (Daletskii and Krein 1965) or Equation 2.40 of (Bhatia 2007)) that states the following in our case: for $\boldsymbol{H} \in T_p\mathcal{P}_d$,

$$\Psi(\boldsymbol{H}) = \left(\exp^{[1]}(\boldsymbol{U})_{ij}\boldsymbol{H}_{ij}\right)_{i,j} \tag{A.1}$$

where

$$\exp^{[1]}(\boldsymbol{U})_{ij} = \begin{cases} e^{\lambda_i} & \text{for } i = j, \\ \frac{e^{\lambda_i} - e^{\lambda_j}}{\lambda_i - \lambda_j} & \text{for } i \neq j. \end{cases}$$

Using the previous formula, one can compute $\Psi(\boldsymbol{E}_{ij}^{I_d})$ for $i \leqslant j$. Now, one needs to compute the coefficients of $\Psi(\boldsymbol{E}_{ij}^{I_d})$ in the basis $(\boldsymbol{E}_{kl}^{\exp(\boldsymbol{U})})_{k\leqslant l}$. As the basis $(\boldsymbol{E}_{kl}^{\exp(\boldsymbol{U})})_{k\leqslant l}$ is orthonormal, one simply needs to compute the dot product between $\Psi(\boldsymbol{E}_{ij}^{I_d})$ and one element of the basis to get the corresponding coefficient. For $k \leqslant l$, one has:

$$\begin{aligned}
\langle \Psi(\boldsymbol{E}_{ij}^{I_d}), \boldsymbol{E}_{kl}^{\exp(\boldsymbol{U})} \rangle_{\exp(\boldsymbol{U})} &= \langle \Psi(\boldsymbol{E}_{ij}^{I_d}), \exp(\boldsymbol{U}/2)\boldsymbol{E}_{kl}^{I_d}\exp(\boldsymbol{U}/2) \rangle_{\exp(\boldsymbol{U})} \\
&\quad \text{(using the definition of } \boldsymbol{E}_{kl}^{\exp(\boldsymbol{U})}\text{)} \\
&= \operatorname{tr}\left(\Psi(\boldsymbol{E}_{ij}^{I_d})\exp(-\boldsymbol{U}/2)\boldsymbol{E}_{kl}^{I_d}\exp(-\boldsymbol{U}/2)\right) \\
&\quad \text{(using the definition of the AIRM metric)} \\
&= \langle \exp(-\boldsymbol{U}/2)\Psi(\boldsymbol{E}_{ij}^{I_d})\exp(-\boldsymbol{U}/2), \boldsymbol{E}_{kl}^{I_d} \rangle_{I_d}
\end{aligned}$$

Therefore, it is the coefficient (k,l) of the matrix $\exp(-\boldsymbol{U}/2)\Psi(\boldsymbol{E}_{ij}^{I_d})\exp(-\boldsymbol{U}/2)$ (up to a factor $\sqrt{2}$ when $k \neq l$). We now need to compute this matrix. As $\boldsymbol{U}$ is diagonal, $\boldsymbol{U} = \operatorname{diag}(\lambda_1, ..., \lambda_d)$, one has that $\exp(-\boldsymbol{U}/2) = \operatorname{diag}(e^{-\lambda_1/2}, ..., e^{-\lambda_d/2})$. Therefore, and using Equation (A.1), the coefficient (k,l) of the matrix $\exp(-\boldsymbol{U}/2)\Psi(\boldsymbol{E}_{ij}^{I_d})\exp(-\boldsymbol{U}/2)$ is:

$$\begin{cases}
e^{-\lambda_i/2}e^{\lambda_i}e^{-\lambda_i/2} = 1 & \text{for } i = j = k = l, \\
\frac{1}{\sqrt{2}}e^{-\lambda_i/2}\frac{e^{\lambda_i} - e^{\lambda_j}}{\lambda_i - \lambda_j}e^{-\lambda_j/2} = \sqrt{2}\frac{\sinh\left(\frac{\lambda_i - \lambda_j}{2}\right)}{\lambda_i - \lambda_j} & \text{for } i \neq j \text{ and } (k,l) = (i,j) \\
0 & \text{if } (k,l) \neq (i,j).
\end{cases}$$

Therefore, the matrix of Ψ in the bases $(\boldsymbol{E}_{ij}^{I_d})_{i\leqslant j}$ and $(\boldsymbol{E}_{kl}^{\exp(\boldsymbol{U})})_{k\leqslant l}$ is diagonal with diagonal coefficients 1 and $2\frac{\sinh\left(\frac{\lambda_i - \lambda_j}{2}\right)}{\lambda_i - \lambda_j}$. Thus, the determinant of Ψ is the product of these coefficients, which gives the result for the diagonal case.

Case 2: $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d$ general Let us now consider the general case where $\boldsymbol{U} \in T_{I_d}\mathcal{P}_d$ is not diagonal. One has that, as $\boldsymbol{U}$ is symmetric, one can diagonalize it: $\boldsymbol{U} = \boldsymbol{G}\Lambda\boldsymbol{G}^\top$ where $\boldsymbol{G}$ is an orthogonal matrix and Λ is diagonal. As $\exp(\boldsymbol{U}) = \boldsymbol{G}\exp(\Lambda)\boldsymbol{G}^\top$, one has that $\mathrm{d}\exp(\boldsymbol{U}) \cdot \boldsymbol{H} = \boldsymbol{G}\left(\mathrm{d}\exp(\Lambda) \cdot (\boldsymbol{G}^\top\boldsymbol{H}\boldsymbol{G})\right)\boldsymbol{G}^\top$. Therefore, as we are only interested in the determinant of $\mathrm{d}\exp(\boldsymbol{U})$, and as $\boldsymbol{G}$ are orthogonal, one has that $\det \mathrm{d}\exp(\boldsymbol{U}) \cdot \boldsymbol{H} = \det\left(\mathrm{d}\exp(\Lambda) \cdot (\boldsymbol{G}^\top\boldsymbol{H}\boldsymbol{G})\right)$. One thus needs to compute $\mathrm{d}\exp(\Lambda) \cdot (\boldsymbol{G}^\top\Lambda\boldsymbol{G})$, and using the Daletskii-Krein formula, as in the diagonal case, one has:

$$\mathrm{d}\exp(\Lambda) \cdot (\boldsymbol{G}^\top\boldsymbol{H}\boldsymbol{G}) = \left(\exp^{[1]}(\boldsymbol{U})_{ij}\tilde{\boldsymbol{H}}_{ij}\right)_{ij} \tag{A.2}$$

where $\exp^{[1]}(\boldsymbol{U})$ is defined as above and $\boldsymbol{G}^\top\boldsymbol{H}\boldsymbol{G} = [[\tilde{\boldsymbol{H}}_{ij}]]$. In order to do the same proof as for the diagonal case, one needs to modify the basis used in $T_{I_d}\mathcal{P}_d$ as in Equation (A.2), the

coefficients of $\boldsymbol{G}^\top \boldsymbol{H} \boldsymbol{G}$ appear (rather than directly the coefficients of $\boldsymbol{H}$ as in the previous case). Therefore, we choose as basis for $T_{I_d}\mathcal{P}_d$ the basis $(\boldsymbol{G}\boldsymbol{E}_{ij}^{I_d}\boldsymbol{G}^\top)_{ij}$. One can easily check that this basis is orthonormal and does not imply any volume changes. One can now use the same proof as for the diagonal case to compute the determinant of $\mathrm{d}\exp(\boldsymbol{U})$, which gives the result for the general case.

For the Jacobian determinant of the exponential mapping at $\boldsymbol{P} \in \mathcal{P}_d$, we recall that the mapping $\phi_{\boldsymbol{P}} \colon \boldsymbol{U} \mapsto \boldsymbol{P}^{-1/2}\boldsymbol{U}\boldsymbol{P}^{-1/2}$ is a linear isometry, so it does not imply any volume changes. Moreover, one has that

$$\phi_{\boldsymbol{P}}(\mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{U})) = \mathrm{Exp}_{I_d}(\phi_{\boldsymbol{P}}(\boldsymbol{U})),$$

thus

$$\mathrm{d}\phi_{\boldsymbol{P}}(\mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{U})) \circ \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{U}) = \mathrm{d}\,\mathrm{Exp}_{I_d}(\phi_{\boldsymbol{P}}(\boldsymbol{U})) \circ \mathrm{d}\phi_{\boldsymbol{P}}(\boldsymbol{U}),$$

which, by taking the determinant, gives us that

$$J_{\boldsymbol{P}}(\boldsymbol{U}) = \det \mathrm{d}\,\mathrm{Exp}_{\boldsymbol{P}}(\boldsymbol{U}) = \det \mathrm{d}\,\mathrm{Exp}_{I_d}(\boldsymbol{P}^{-1/2}(\boldsymbol{U})\boldsymbol{P}^{-1/2}) = J_{I_d}(\boldsymbol{P}^{-1/2}(\boldsymbol{U})\boldsymbol{P}^{-1/2}).$$

□

A.2.2 The identifiability of the wrapped Gaussian on $\mathbb{H}^d$

Theorem 3.42 (Identifiability of the wrapped Gaussian on the hyperbolic space)**.** *The wrapped Gaussian* ***is identifiable*** *on the hyperbolic space, meaning that*

$$\mathrm{WG}(p;\mu,\Sigma) = \mathrm{WG}(q;\nu,\Gamma) \iff p = q, \mu = \nu \text{ and } \Sigma = \Gamma.$$

Proof. For this proof, we consider the Poincaré ball model $\mathbb{B}^d$. We refer to Mathieu et al. (2019) for a detailed presentation of this model and for the expressions of the Riemannian exponential and logarithmic mappings. Similarly to the proof of Theorem 3.18 in the case of $\mathcal{P}_d$, it is enough to show the result for $\mathrm{WG}(0_d;\mu,\Sigma)$. The sufficient condition ($\Leftarrow$) is evident, so let us focus on the necessary one ($\Rightarrow$). Let us suppose that $\mathrm{WG}(0_d;\mu,\Sigma) = \mathrm{WG}(q;\nu,\Gamma)$. This is equivalent to the equality of the densities that can be rewritten as:

$$\forall u \in T_{0_d}\mathbb{H}^d,\ \frac{g_{\mu,\Sigma}(\mathrm{Vect}(u))}{|J(u)|} = \frac{g_{\nu,\Gamma}(\mathrm{Vect}_q(\mathrm{Log}_q(\mathrm{Exp}_{0_d}(u))))}{|J_q(\mathrm{Log}_q(\mathrm{Exp}_{0_d}(u)))|}, \tag{A.3}$$

where we denote $\mathrm{Vect} = \mathrm{Vect}_{0_d}$ and $J = J_{0_d}$. The main idea is to study the limits when $\|u\| \to \infty$ of both terms of this equality. We will show that they do not have the same limit if $q \neq 0_d$. Let us suppose that $q \neq 0_d$. Then, using Equation (11) of Nagano et al. (2019), for the Jacobian determinant of the exponential mapping Exp_{0_d}, one has that

$$J(u) = \left(\frac{e^{\|u\|} - e^{-\|u\|}}{2\|u\|}\right)^{d-1} \sim \frac{e^{(d-1)\|u\|}}{(2\|u\|)^{d-1}} \quad (\|u\| \to \infty).$$

Thus, we can show, using the expression of the density of a multivariate Gaussian $g_{\mu,\Sigma}$, that

$$\frac{g_{\mu,\Sigma}(\mathrm{Vect}(u))}{|J(u)|} \xrightarrow[\|u\|\to\infty]{} 0.$$

Now focus on the right-hand side of Equation (A.3), compute that $\mathrm{Exp}_{0_d}(u) = \tanh(\|u\|)\frac{u}{\|u\|}$, plug it into the expression of the Möbius addition given in Mathieu et al. (2019), and get that

$$-q \oplus \mathrm{Exp}_{0_d}(u) = \frac{-\left(1 + 2\frac{\tanh(\|u\|)}{\|u\|}\langle -q, u\rangle + \tanh(\|u\|)^2\right) q + (1 - \|q\|)\tanh(\|u\|)\frac{u}{\|u\|}}{1 + 2\frac{\tanh(\|u\|)}{\|u\|}\langle -q, u\rangle + \|q\|^2\|u\|^2}.$$

As tanh is bounded between -1 and 1 and $\frac{u}{\|u\|}$ is restricted to the unit circle, one can show that the numerator is a bounded function of u. Additionally, as $q \neq 0_d$, the denominator is equivalent to $\|q\|^2\|u\|^2$ and one can see that $-q \oplus \mathrm{Exp}_{0_d}(u) \xrightarrow[\|u\|\to\infty]{} 0$. Finally, using that $\frac{\tanh^{-1}(t)}{t} \xrightarrow[t\to 0]{} 1$, we have that $\mathrm{Log}_q(\mathrm{Exp}_{0_d}(u)) \xrightarrow[\|u\|\to\infty]{} 0$. The density of the multivariate Gaussian $g_{\nu,\Gamma} \circ \mathrm{Vect}_q$ and the Jacobian determinant J_q being continuous, one has the limit of the right-hand side of Equation (A.3) when $\|u\| \to \infty$:

$$\frac{g_{\nu,\Gamma}(\mathrm{Vect}_q(\mathrm{Log}_q(\mathrm{Exp}_{0_d}(u))))}{|J_q(\mathrm{Log}_q(\mathrm{Exp}_{0_d}(u)))|} \xrightarrow[\|u\|\to\infty]{} \frac{g_{\nu,\Gamma}(0_d)}{|J_q(0)|} = \frac{1}{(2\pi \det \Gamma)^{d/2}} e^{-\frac{1}{2}\nu^\top \Gamma \nu} \neq 0.$$

This is a contradiction, and we can conclude that $q = 0_d$. We conclude by using the identifiability of the multivariate Gaussian $\mathcal{N}(\mu, \Sigma)$ to show that $\nu = \mu$ and $\Gamma = \Sigma$. □

A.3 Proofs of Chapter 5

A.3.1 The proof of the general conditional vector field

In this section, we give a proof of Theorem 5.5, which we start by recalling.

Theorem 5.5. *Let $p_t(\cdot \mid \boldsymbol{Z})$ be the conditional probability path given in Equation (5.9) and Φ_t its flow given in Equation (5.10). The vector field defining Φ_t depends on the value of $\sigma_t(\boldsymbol{Z})$ in the following manner:*

- *If $\sigma_t(\boldsymbol{Z}) > 0$, then the unique vector field defining Φ_t is*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y} + \boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \bar{\boldsymbol{X}}_t^{1/2}\,\mathrm{d}\exp(\log \boldsymbol{M}_t) \cdot \left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{M}_t - \mu_t) + \mu_t'\right]\bar{\boldsymbol{X}}_t^{1/2},$$

  where $\boldsymbol{S}$ is the derivative of $t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}$, $\boldsymbol{M}_t = \bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}$, and all time-dependent quantities are evaluated at $\boldsymbol{Z}$.
- *If $\sigma_t(\boldsymbol{Z}) = 0$, then*

$$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t^{1/2}\exp(\mu_t)\bar{\boldsymbol{X}}_t^{1/2}\right],$$

  where all time-dependent quantities are evaluated at $\boldsymbol{Z}$.

Proof. We begin by recalling the conditional flow Φ_t from Equation (5.10):

$$\Phi_t(\boldsymbol{X}) = \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp\left[\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right]\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}.$$

Let us differentiate this flow with respect to t. For this, we denote

$$\begin{cases} \boldsymbol{S} \text{ the derivative of } t \mapsto \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \\ g(t) = \frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z}). \end{cases}$$

Then, one has

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) = \boldsymbol{S}(t)\exp\left[g(t)\right]\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} + \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\mathrm{d}\exp(g(t))\cdot[g'(t)]\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} + \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp\left[g(t)\right]\boldsymbol{S}(t).$$

Then, one can compute that

$$g'(t) = \frac{\sigma_t'(\boldsymbol{Z})}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + \mu_t'(\boldsymbol{Z}).$$

Thus, going back to $\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X})$:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) &= \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\Phi_t(\boldsymbol{X}) + \Phi_t(\boldsymbol{X})\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \\ &\quad \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\mathrm{d}\exp\left(\frac{\sigma_t(\boldsymbol{Z})}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + \mu_t(\boldsymbol{Z})\right)\cdot\left[\frac{\sigma_t'(\boldsymbol{Z})}{\sigma_{\text{init}}}\log(\boldsymbol{X}) + \mu_t'(\boldsymbol{Z})\right]\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2} \\ &= u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}).\end{aligned}$$

Let us denote $\boldsymbol{Y} = \Phi_t(\boldsymbol{X})$. We now need to distinguish between the values of σ_t.

- **If** $\sigma_t > 0$, then Φ_t is invertible and $\boldsymbol{X} = \exp\left(\frac{\sigma_{\text{init}}}{\sigma_t(\boldsymbol{Z})}(\log \boldsymbol{M}_t - \mu_t(\boldsymbol{Z}))\right)$ where $\boldsymbol{M}_t = \bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}$. Plugging this into the previous equation, we get:

  $$u_t(\boldsymbol{Y} \mid \boldsymbol{Z}) = \boldsymbol{S}(t)\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{Y} + \boldsymbol{Y}\bar{\boldsymbol{X}}_t^{-1/2}\boldsymbol{S}(t) + \bar{\boldsymbol{X}}_t^{1/2}\,\mathrm{d}\exp(\log \boldsymbol{M}_t)\cdot\left[\frac{\sigma_t'}{\sigma_t}(\log \boldsymbol{M}_t - \mu_t) + \mu_t'\right]\bar{\boldsymbol{X}}_t^{1/2}.$$

- **If** $\sigma_t = 0$, then Φ_t is not invertible. Going back to the general form of Φ_t given at Equation (5.10), we have that

  $$\Phi_t(\boldsymbol{X}) = \bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp(\mu_t(\boldsymbol{Z}))\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}.$$

  Thus, the ODE is now simply

  $$\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t(\boldsymbol{X}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp(\mu_t(\boldsymbol{Z}))\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\right] = u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}),$$

  and thus $u_t(\Phi_t(\boldsymbol{X}) \mid \boldsymbol{Z}) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\exp(\mu_t(\boldsymbol{Z}))\bar{\boldsymbol{X}}_t(\boldsymbol{Z})^{1/2}\right]$.

Thus, $u_t(\cdot \mid \boldsymbol{Z})$ generates the wrapped Gaussian path $p_t(\cdot \mid \boldsymbol{Z})$. □

A.4 Proofs of Chapter 7

A.4.1 Proof of the Riemannian gradient of the t-SNE

Proposition 7.1 (Riemannian gradient of the t-SNE cost)**.** *The Riemannian gradient of the t-SNE cost defined as Equation* (D.5) *is given by:*

$$\nabla_{\boldsymbol{Y}_i}C = -4\sum_{j=1}^{N}\frac{p_{ij} - q_{ij}}{1 + \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2}\operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j).$$

Proof. We recall the following notations:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2} \quad q_{ij} = \frac{(1 + \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2)^{-1}}{\sum_{k\neq l}(1 + \delta(\boldsymbol{Y}_k, \boldsymbol{Y}_l)^2)^{-1}}.$$

We want the gradient with respect to Y_i of the following cost function:

$$C = \sum_{i=1}^{N}\mathrm{KL}(P_i||Q_i) = \sum_{i=1}^{N}\sum_{j\neq i}p_{ij}\log\frac{p_{ij}}{q_{ij}}.$$

This gradient is given by:

$$\nabla_{\boldsymbol{Y}_i}C = -4\sum_{j=1}^{N}\frac{p_{ij} - q_{ij}}{1 + \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2}\operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j).$$

Let us define the following variables to facilitate the computations:

$$d_{ij} = \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)$$
$$Z = \sum_{k \neq l} (1 + \delta(\boldsymbol{Y}_k, \boldsymbol{Y}_l)^2)^{-1}$$

Therefore, we have $q_{ij} = \frac{(1+d_{ij}^2)^{-1}}{Z}$. To compute the gradient $\nabla_{\boldsymbol{Y}_i} C$, we use the chain rule and the symmetry of $d_{ij} = d_{ji}$:

$$\nabla_{\boldsymbol{Y}_i} C = 2 \sum_j \frac{\partial C}{\partial d_{ij}} \nabla_{\boldsymbol{Y}_i} d_{ij}. \tag{A.4}$$

Using the computation of the gradient of the Riemannian MDS and especially the Equation (7.4), we have

$$\nabla_{\boldsymbol{Y}_i} d_{ij} = -\frac{\operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j)}{d_{ij}}.$$

Therefore, we will focus on the computation of $\frac{\partial C}{\partial d_{ij}}$. Using the notations defined above, we have

$$C = \sum_{i=1}^{N} \sum_{j \neq i} p_{ij} \log p_{ij} - \sum_{i=1}^{N} \sum_{j \neq i} p_{ij} \log \left(\frac{(1 + d_{ij}^2)^{-1}}{Z} \right).$$

The first part of this sum does not depend on d_{ij}, thus, we have

$$\begin{aligned} \frac{\partial C}{\partial d_{ij}} &= -\sum_{k=1}^{N} \sum_{l \neq k} p_{kl} \frac{\partial}{\partial d_{ij}} \log \left(\frac{(1 + d_{kl}^2)^{-1}}{Z} \right) \\ &= -\sum_{k=1}^{N} \sum_{l \neq k} p_{kl} \frac{\partial}{\partial d_{ij}} \left(-\log Z - \log(1 + d_{kl}^2) \right) \\ &= \sum_{k=1}^{N} \sum_{l \neq k} p_{kl} \frac{\partial}{\partial d_{ij}} \log Z + \sum_{k=1}^{N} \sum_{l \neq k} p_{kl} \frac{\partial}{\partial d_{ij}} \log(1 + d_{kl}^2) \end{aligned} \tag{A.5}$$

Let us compute those two terms separately:

- For the first term, let us start by computing $\frac{\partial}{\partial d_{ij}} \log Z$:

$$\begin{aligned} \frac{\partial}{\partial d_{ij}} \log Z &= \frac{1}{Z} \frac{\partial}{\partial d_{ij}} Z \\ &= \frac{1}{Z} \frac{\partial}{\partial d_{ij}} \sum_{k \neq l} (1 + \delta(\boldsymbol{Y}_k, \boldsymbol{Y}_l)^2)^{-1} \quad \text{by definition of } Z \\ &= -\frac{1}{Z} \frac{2 d_{ij}}{(1 + d_{ij}^2)^2} \quad \text{as the only non-zero term of the derivative of the sum is when } k{=}i \text{ and } l{=}j. \\ &= -2 \frac{q_{ij}}{(1 + d_{ij}^2)} d_{ij} \quad \text{using the fact that } q_{ij}{=}\tfrac{(1+d_{ij}^2)^{-1}}{Z}. \end{aligned}$$

  Therefore,

$$\sum_{k=1}^{N} \sum_{l \neq k} p_{kl} \frac{\partial}{\partial d_{ij}} \log Z = -2 q_{ij} (1 + d_{ij}^2)^{-1} d_{ij} \underbrace{\sum_{k=1}^{N} \sum_{l \neq k} p_{kl}}_{=1} = -2 q_{ij} (1 + d_{ij}^2)^{-1} d_{ij}.$$

- For the second term, we start by computing $\frac{\partial}{\partial d_{ij}} \log(1 + d_{kl}^2)$:

$$\frac{\partial}{\partial d_{ij}} \log(1 + d_{kl}^2) = \frac{1}{(1 + d_{kl}^2)} \frac{\partial}{\partial d_{ij}} (1 + d_{kl}^2) = \begin{cases} 2 d_{ij} (1 + d_{ij}^2)^{-1} & \text{if } k = i \text{ and } l = j \\ 0 \quad \text{otherwise} \end{cases}$$

Thus,

$$\sum_{k=1}^{N}\sum_{l\neq k} p_{kl}\frac{\partial}{\partial d_{ij}}\log(1+d_{kl}^2) = 2d_{ij}(1+d_{ij}^2)^{-1}p_{ij}.$$

Stitching all together in Equation (A.5), we have

$$\frac{\partial C}{\partial d_{ij}} = -2q_{ij}(1+d_{ij}^2)^{-1}d_{ij} + 2d_{ij}(1+d_{ij}^2)^{-1}p_{ij} = 2d_{ij}(p_{ij}-q_{ij})(1+d_{ij}^2)^{-1}.$$

We can now go back to Equation (A.4), and we have

$$\nabla_{\boldsymbol{Y}_i} C = 2\sum_j 2d_{ij}(p_{ij}-q_{ij})(1+d_{ij}^2)^{-1}\left(-\frac{\operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j)}{d_{ij}}\right) = -4\sum_j (p_{ij}-q_{ij})(1+d_{ij}^2)^{-1}\operatorname{Log}_{\boldsymbol{Y}_i}(\boldsymbol{Y}_j).$$

□

Appendix B

Appendix to Chapter 5

Contents

B.1 More details on the synthetic experiments 235
B.1.1 Details on the datasets 235
B.1.2 Details on the architecture used 236
B.2 Details on the training and sampling times 237
B.3 More details on real-world experiments 238
B.3.1 Datasets preprocessing 238
B.3.2 Training details 238

B.1 More details on the synthetic experiments

In this section, we give more details on the synthetic experiments we conducted in Section 5.5. We start by detailing the different datasets used (*Swiss-Roll*, *Figure-8* and *Mixture of WG*) and then give the setup and the architecture used.

For all of those three datasets, the training dataset is composed of 8000 matrices, the validation set is composed of 1000 matrices and the testing set is composed of 1000 matrices. We created 5 different versions of each dataset, each sampled with a seed ranging from 0 to 4.

B.1.1 Details on the datasets

The 2×2 SPD Swiss-Roll We give here the equations that were used to generate the SPD Swiss-roll dataset. We start by generating a 3D spiral using the following equations:

$$u = t\cos(t), \quad v = t\sin(t), \quad w = t + s + 1, \tag{B.1}$$

where $t \sim \mathcal{U}[0, 2\pi]$ is the angle of the spiral and $s \sim \mathcal{U}[0, 3]$ is the ribbon's width. Then, we rotate the 3D spiral to put it into the cone of equation $a > 0$ and $ac > b^2$ i.e. the cone of 2×2 SPD matrices of the form $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$:

$$a = \frac{1}{2}(w + u), \quad c = \frac{1}{2}(w - u), \text{ and } b = \frac{1}{2}v. \tag{B.2}$$

Thus, the matrix

$$\boldsymbol{X} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

will be SPD and will form a spiral in the cone of 2×2 SPD matrices.

The Figure-8 Now, let us give the equations used to create the SPD Figure-8 dataset. We use a technique similar to the one used for the above Swiss-Roll dataset. We start by creating a lemniscate-like shape using the following equations:

$$u = \cos(t), \quad v = \sin(2t), \quad w = t + s + \frac{1}{2}, \tag{B.3}$$

where $t \sim \mathcal{U}[0, 2\pi]$ is the angle of the spiral and $s \sim \mathcal{U}[0, 1]$ is the ribbon's width. Then, we rotate the 3D spiral to put it into the cone of equation $a > 0$ and $ac > b^2$ i.e. the cone of 2×2 SPD matrices of the form $\begin{pmatrix} a & b \\ b & c \end{pmatrix}$:

$$a = \frac{1}{2}(w + u), \quad c = \frac{1}{2}(w - u), \text{ and } b = \frac{1}{2}v. \tag{B.4}$$

Thus, the matrix

$$\boldsymbol{X} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

will be SPD and will form an 8 shape in the cone of 2×2 SPD matrices.

The Mixture of WG The last dataset is composed of 10×10 SPD matrices sampled from a mixture of three wrapped Gaussians. The mixture is:

$$q = \frac{1}{3}\mathrm{WG}(\boldsymbol{X}_1; 0_{55}, \sigma^2_{\text{init}} I_{55}) + \frac{1}{3}\mathrm{WG}(\boldsymbol{X}_2; 0_{55}, \sigma^2_{\text{init}} I_{55}) + \frac{1}{3}\mathrm{WG}(\boldsymbol{X}_3; 0_{55}, \sigma^2_{\text{init}} I_{55}).$$

Here, the dataset has a simple shape. However, the goal is to produce matrices with different condition numbers. We recall that, for an SPD matrix $\boldsymbol{X}$, the condition number $\kappa(\boldsymbol{X})$ can be computed as

$$\kappa(\boldsymbol{X}) = \frac{\lambda_{\max}(\boldsymbol{X})}{\lambda_{\min}(\boldsymbol{X})},$$

where $\lambda_{\max}(\boldsymbol{X})$ is the largest eigenvalue of $\boldsymbol{X}$ and $\lambda_{\min}(\boldsymbol{X})$ is the smallest (see Section 2.6.2 of Golub and Van Loan 2013). When this condition number is very large compared to one, the matrix $\boldsymbol{X}$ is said to be *ill-conditioned*, and the computation of its inverse, or solution of a linear system of equations is prone to large numerical errors.

In this experiment, we wanted to generate the centers of the components of the wrapped Gaussian mixture to have specific condition numbers $\kappa \in \{10, 10^2, 10^3, 10^4\}$. For each experiment, we set a specific κ. Then, the centers $\boldsymbol{X}_k$ are chosen so that they all have as condition number κ. In order to do so, we choose $\boldsymbol{X}_k$ as:

$$\boldsymbol{X}_k = \boldsymbol{Q}_k \operatorname{diag}(\lambda_1, ..., \lambda_d) \boldsymbol{Q}_k^T$$

with the eigenvalues $(\lambda_1, ..., \lambda_d)$ log-spaced in $[\kappa^{-1/2}, \kappa^{1/2}]$ and an independent random orientation $\boldsymbol{Q}_k$. The matrices $\boldsymbol{X}_k$ have, by construction, exactly a condition number equal to κ. The dataset is centered, and we sample the same amount of points from each of the three wrapped Gaussians, and label them according to the component of the mixture we sampled from.

B.1.2 Details on the architecture used

For the two first 2×2 datasets (the 2×2 *SPD Swiss-roll* and the *Figure-8*), the conditional vector field $u_t(\cdot \mid \boldsymbol{Z})$ was learned using an MLP composed of 6 layers, each composed of 128 neurons and a SELU activation function (*Scaled Exponential Linear Unit*, Klambauer et al. 2017). Timestep and class information are injected by concatenating them to the input. The

input $\boldsymbol{X}$ of conditional vector field $u_t(\boldsymbol{X} \mid \boldsymbol{Z})$ is an SPD matrix and the output is an element of the tangent space $T_{\boldsymbol{X}}\mathcal{P}_d$; they are both symmetric matrices. Therefore, we vectorized the input matrix by taking only the elements on the upper-triangular part and unvectorizing the output matrix.

For the higher dimensional dataset, we used a more complex architecture composed of 10 layers each with 512 hidden units and a SELU activation function.

For all the synthetic experiments, we optimized the model using the AdamW optimizer (Loshchilov and Hutter 2017) with default $\beta = (0.9, 0.999)$ and $\epsilon = 10^{-8}$ with weight decay explicitly disabled. The base learning rate is 10^{-3}, modulated by a two-phase schedule stepped once per epoch. It is built using a linear warmup going from 10^{-4} to 10^{-3} during the first 10 epochs and cosine annealing. Gradients are clipped to global norm 1.0, a guard against the loss spikes observed at high condition number. An exponential moving average on the weights (B. T. Polyak and Juditsky 1992) was used with a decay of 0.999. We trained it for 12000 epochs. The synthetic experiments were conducted on an NVIDIA A40 GPU with 40GB of memory. We train our Riemannian models with the AIRM loss and the Euclidean CFM using the classical Frobenius loss on symmetric matrices.

Importantly, all the training was done in `float64`, the computations on SPD matrices requiring more precision.

B.2 Details on the training and sampling times

Dataset	DiffeoCFM	EuclideanCFM	RiemannianCFM	WrappedCFM
Figure-8	$23.2_{\pm 0.4}$	$\mathbf{18.3}_{\pm 1.5}$	$24.2_{\pm 0.2}$	$26.8_{\pm 1.1}$
Swiss-roll	$21.6_{\pm 1.4}$	$\mathbf{17.6}_{\pm 0.5}$	$24.2_{\pm 0.4}$	$24.7_{\pm 0.2}$
WG mixture $\kappa = 10$	$91.0_{\pm 8.9}$	$\mathbf{78.3}_{\pm 5.0}$	$104.4_{\pm 3.6}$	$115.8_{\pm 7.3}$
WG mixture $\kappa = 10^2$	$94.2_{\pm 9.8}$	$\mathbf{75.8}_{\pm 4.5}$	$108.1_{\pm 5.0}$	$120.6_{\pm 5.6}$
WG mixture $\kappa = 10^3$	$\mathbf{46.8}_{\pm 13.6}$	$53.3_{\pm 4.6}$	$59.7_{\pm 4.5}$	$63.8_{\pm 4.7}$
WG mixture $\kappa = 10^4$	$62.1_{\pm 13.6}$	$\mathbf{50.2}_{\pm 0.2}$	$63.3_{\pm 4.5}$	$66.3_{\pm 4.8}$

Table B.1. Mean training time per synthetic dataset and method (minutes, mean ± std).

Training times In this section, we report on the training time of the different methods on the different datasets. For the synthetic experiments, we give the results in Table B.1. We can see that our proposed method (WrappedCFM) is the slowest to train, which is coherent as the conditional vector field $u_t(\cdot \mid \boldsymbol{X}_1)$ is the most complicated to compute (see Theorem 5.6) and it is computed at each iteration of the training.

Dataset	DiffeoCFM	EuclideanCFM	RiemannianCFM	WrappedCFM
Figure-8	$0.121_{\pm 0.004}$	$\mathbf{0.115}_{\pm 0.001}$	$0.279_{\pm 0.002}$	$0.280_{\pm 0.003}$
Swiss-roll	$0.116_{\pm 0.000}$	$\mathbf{0.113}_{\pm 0.000}$	$0.278_{\pm 0.001}$	$0.278_{\pm 0.001}$
WG mixture $\kappa = 10$	$1.271_{\pm 0.001}$	$\mathbf{1.260}_{\pm 0.000}$	$2.532_{\pm 0.100}$	$2.480_{\pm 0.005}$
WG mixture $\kappa = 10^2$	$1.270_{\pm 0.001}$	$\mathbf{1.261}_{\pm 0.000}$	$2.472_{\pm 0.010}$	$2.470_{\pm 0.011}$
WG mixture $\kappa = 10^3$	$1.270_{\pm 0.001}$	$\mathbf{1.260}_{\pm 0.000}$	$2.453_{\pm 0.009}$	$2.510_{\pm 0.112}$
WG mixture $\kappa = 10^4$	$1.287_{\pm 0.033}$	$\mathbf{1.261}_{\pm 0.001}$	$2.438_{\pm 0.006}$	$2.439_{\pm 0.005}$

Table B.2. Mean sampling time for 1000 samples per dataset and method (seconds, mean ± std).

Sampling times We also report the sampling times for 1000 samples generated using the different methods in Table B.2. Coherently, the EuclideanCFM is the fastest method to

sample from, as it only relies on a Euclidean Euler scheme. Then comes the DiffeoCFM, for which a Riemannian logarithm and a Riemannian exponential are added to the computation. For the RiemannianCFM and the WrappedCFM, they both rely on a Riemannian Euler schema, as described in Equation (5.15). Thus, they are slower to compute. Nevertheless, all methods have very reasonable sampling times, including WrappedCFM.

B.3 More details on real-world experiments

B.3.1 Datasets preprocessing

BCI

The BCI datasets that we consider are *BNCI2014002* (Steyrl et al. 2016) and *BNCI2015001* (Faller et al. 2012). They consist of several subjects and several sessions per subject performing a Motor Imagery task (G. Pfurtscheller and Lopes da Silva 1999). We used the library MOABB (Aristimunha, Carrara, et al. 2026) to load and preprocess the data. The preprocessing consists of a band-pass filter in the range $[7, 35]$ Hz. Then, we used the OAS estimator (Y. Chen et al. 2010) to compute the covariance matrices. We follow a leave-one-group-out protocol. BNCI2014-002 comprises 5 groups, and BNCI2015-001 comprises 2 groups. We followed the same preprocessing as in Collas, Ju, et al. (2026).

Textile

This dataset is made of a set of real images from textile manufacturing that contain non-defective and defective woven textiles. These images come from the public MVTec Anomaly Detection dataset (Bergmann et al. 2021). The same preprocessing as in (Smith, Laubach, et al. 2022) was applied to get the covariance matrices. The two classes correspond to defective and non-defective textiles.

BreizhCrops

This dataset is intended for supervised classification of field crops from satellite time series. We use the default geographical split, where regions frh01–frh03 form the training set and frh04 the test set. Since the dataset is strongly imbalanced, we select the 4 most represented classes. More details on this dataset can be found in the original paper (Rußwurm et al. 2020).

affine-invariant metric.

B.3.2 Training details

Hyperparameters The datasets are split into train, validation, test with 80/10/10 ratio. All the models are trained using the same MLP architecture as the synthetic experiments, with 6 layers and 512 hidden units. We trained for 200k iterations on BreizhCrops and 100k iterations for all other datasets. We used the Adam optimizer with a learning rate of 10^{-4} and exponential moving average on the weights with a decay of 0.999 for all our experiments. All the experiments are repeated 5 times with different random seeds.

Classifier The classifier is an ℓ_2-regularized logistic regression classifier trained on SPD matrices that have been projected onto the tangent space at their Fréchet mean. The regularization parameter of the classifier is selected by 5-fold cross-validation.

Appendix C

Appendix to Chapter 7

Contents

C.1 Effect of the perplexity on the Riemannian t-SNE 239
C.2 More plots on the influence of the sensors 240
C.2.1 Using an MDM classifier 240
C.2.2 Using a Tangent Space LDA 241

C.1 Effect of the perplexity on the Riemannian t-SNE

In this section, we discuss the perplexity parameter that plays an important role in the t-SNE algorithm. We recall that the perplexity is a hyperparameter chosen by the user that will be used to compute the variance of the Gaussians in the conditional probabilities defined in Equation (7.1). The role of the perplexity is to fix the entropy of the conditional probabilities. As explained in Van Assel et al. (2023), the perplexity can be interpreted as the effective number of neighbors for each data point. It is a way of balancing attention between some local and some global aspects of the data one wants to reduce. It is known that varying this parameter can change a lot the results one gets from the t-SNE, and there is no silver bullet method to choose the correct perplexity (Wattenberg et al. 2016).

In the Euclidean case, Maaten and Hinton (2008) say that "The performance of SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50". We want to see if this statement is still true in the Riemannian case. For this, we used the Riemannian t-SNE algorithm with different values of the perplexity $\texttt{perp} \in [5, 288]$ for the covariance matrices estimated from the EEGs of subject 8 of BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). This dataset is composed of 288 SPD matrices of size 22×22. The initial point of the optimization process is fixed to only see the influence of the perplexity. We show the results in Figure C.1. We see that the perplexity has an important impact on the result. When the perplexity is too small, the points further from the origin have huge coefficients ($\simeq 10^{27}$) and are all aligned. As the perplexity grows, the coefficients of the points get smaller and smaller, and we are able to see some structure in the dataset appear. Contrary to the Euclidean t-SNE, we need a larger perplexity to have a convincing result. However, when the perplexity is too big, here where it is equal to 288, all the point cloud collapses to a sphere, without any structure. Throughout our experiments, we found empirically that taking the perplexity to be equal to 3/4 of the total number of points works well.

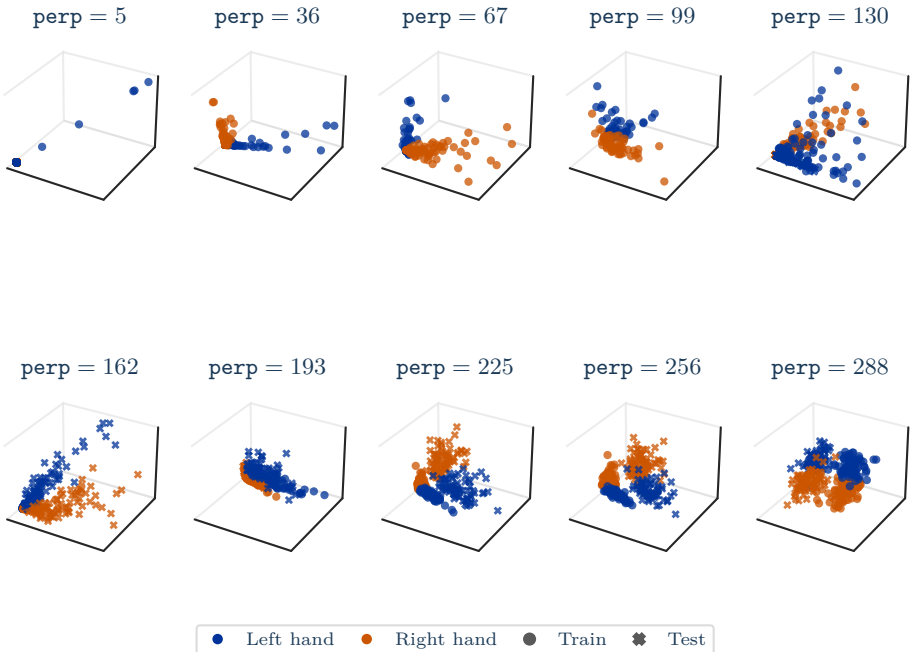


Figure C.1. Effect of changing the perplexity in the Riemannian t-SNE algorithm on the final embedding. The dataset is composed of covariance matrices estimated from the EEGs of subject 8 of BNCI2014001 (Tangermann, K.-R. Müller, et al. 2012). This dataset is composed of 288 SPD matrices of size 22×22.

C.2 More plots on the influence of the sensors

C.2.1 Using an MDM classifier

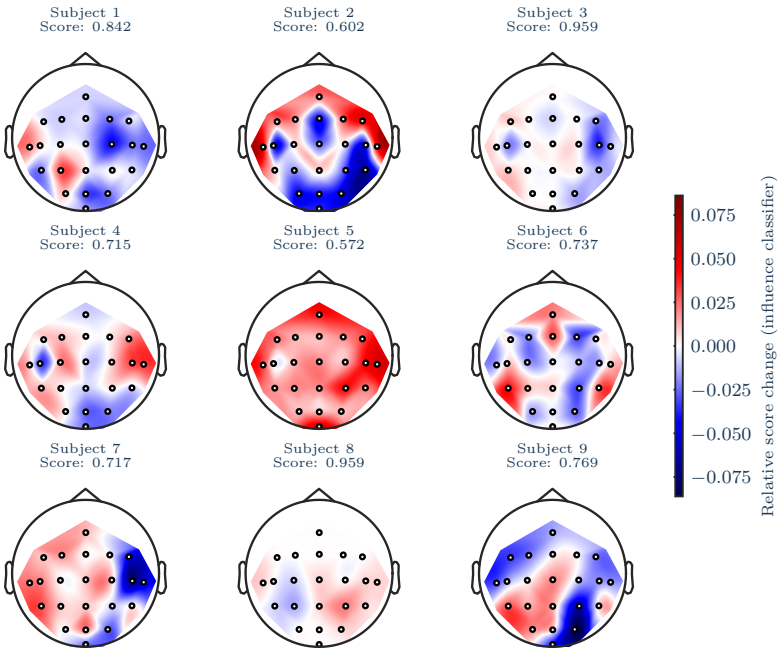


Figure C.2. Result of removing an electrode for all the subjects of BNCI2014001 using an MDM classifier.

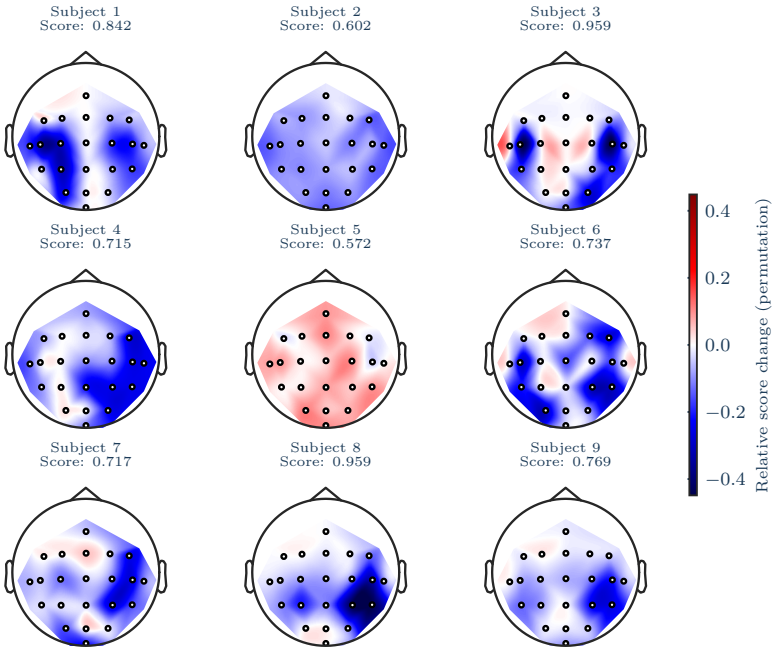


Figure C.3. Result of Permutation Feature Importance for all the subjects of BNCI2014001 using an MDM classifier

C.2.2 Using a Tangent Space LDA

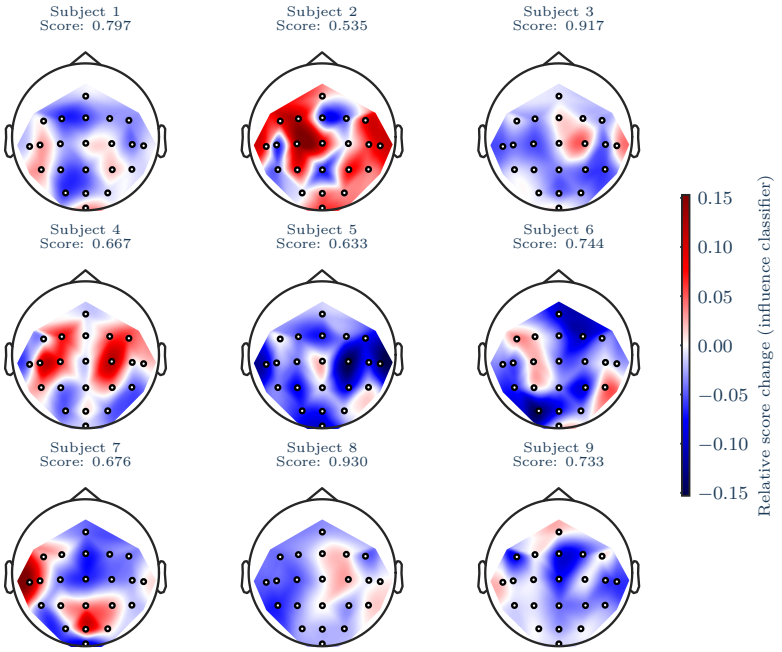


Figure C.4. Result of removing an electrode for all the subjects of BNCI2014001 using a Tangent Space LDA.

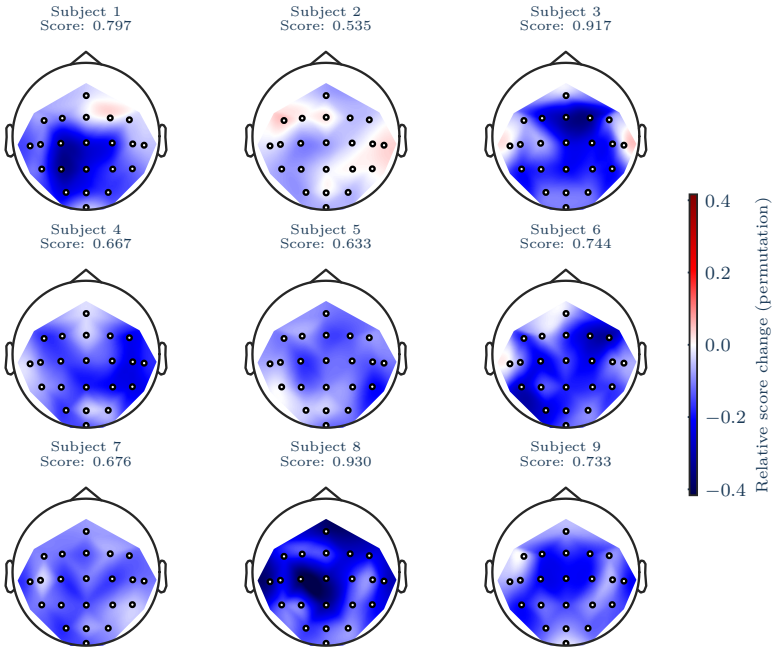


Figure C.5. Result of Permutation Feature Importance for all the subjects of BNCI2014001 using a Tangent Space LDA.

Appendix D

Résumé détaillé en français

D.1 Introduction

De nombreux algorithmes de classification automatique se basent sur la quantification de la similarité entre différentes données. L'hypothèse sous-jacente est que des points proches auront tendance à partager des propriétés communes. Quantifier une similarité entre différents points peut être vu comme un problème géométrique qui consiste à mesurer des distances entre points. Afin de mesurer correctement la distance entre deux points, il est essentiel de considérer la bonne géométrie. En effet, comme illustré à la figure D.1 dans le cas de la sphère $\mathcal{S}^2$ en trois dimensions, mesurer la longueur de la ligne droite entre les points x et y revient à traverser la sphère, ce qui n'est pas physiquement réaliste. Afin de régler ce problème, les mathématiciens du XIX^{e} siècle ont développé de nouvelles géométries (Jenkovszky et al. 2023 ; Riemann 1867) capables de prendre en compte les structures des données. Une de ces géométries, la *géométrie de Riemann*, sera au cœur de ces travaux. Cette géométrie permet de prendre en compte une éventuelle courbure de l'espace et nous fournit des outils pour mesurer des distances, des angles et des volumes.

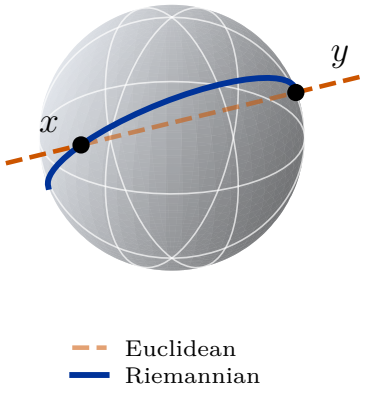


Figure D.1. La sphère $\mathcal{S}^2$. Il faut prendre en compte sa géométrie courbe pour tracer un chemin cohérent entre deux points x et y sur la sphère.

Alors que les mathématiciens ont bien compris l'importance du choix de la bonne géométrie pour des données structurées, les informaticiens ont construit la majorité de leurs algorithmes en faisant l'hypothèse que les données étaient euclidiennes et donc plates. Comme le note Papillon et al. (2025), cela fait seulement deux décennies que le domaine de l'apprentissage automatique commence à voir au-delà des espaces plats. C'est tout l'enjeu des *statistiques géométriques*, domaine dans lequel se place cette thèse.

La motivation principale : les Interfaces Cerveau-Ordinateur

La motivation et l'application principale de ces travaux sont les Interfaces Cerveau-Ordinateur (ICO). L'objectif d'une ICO est de traduire l'activité du cerveau en commandes concrètes pour une application interactive (Clerc et al. 2016). Cela commence par enregistrer l'activité

électrique du cerveau, et dans le cadre de cette thèse, nous nous focaliserons sur l'électroencéphalographie, qui consiste à placer des électrodes sur le crâne d'un sujet, chaque électrode enregistrant l'activité électrique d'une certaine zone du cerveau. Cet enregistrement est appelé un ÉlectroEncéphaloGramme (EEG), qui est une série temporelle multivariée. L'objectif d'une ICO est donc de traduire cet EEG en une commande concrète. Dans cette thèse, nous estimons, à partir d'un EEG, sa matrice de covariance, et c'est cette caractéristique que nous allons classifier (Barachant, Bonnet et al. 2010b ; Congedo et al. 2017 ; Yger, Berar et al. 2017). Cette caractéristique est très structurée mathématiquement, car les matrices de covariance sont des matrices Symétriques Définies Positives (SDP), dont nous notons l'ensemble $\mathcal{P}_d$. Comme pour les points de la sphère $\mathcal{S}^2$, il faut employer une géométrie particulière pour manipuler correctement ces matrices. Dans ces travaux, nous utiliserons la géométrie riemannienne affine-invariante (Pennec 2020).

Problématique

Le paradigme actuel de l'apprentissage statistique et automatique consiste à modéliser les données à l'aide d'une distribution de probabilité (Vapnik 2013 ; Hastie et al. 2009). Ainsi, un ensemble de matrices de covariance estimées à partir de vrais EEG doit pouvoir être modélisé par une distribution de probabilité. Cette distribution doit prendre en compte la géométrie particulière de l'ensemble des matrices SDP. La problématique principale de cette thèse est donc la suivante :

> *Question 4.* Quels modèles statistiques tenant compte de la géométrie fournissent des représentations utiles sur la variété des matrices SDP ?

Cette question principale appelle deux autres sous-questions. La première concerne l'utilisation de ce modèle statistique pour construire de nouveaux outils d'apprentissage :

> *Question 5.* Ce modèle statistique peut-il contribuer à la conception d'outils d'apprentissage plus efficaces pour les matrices SDP ?

Puis, la dernière question se concentre sur l'application de ces travaux aux ICO et sur l'une de leurs principales difficultés : les variabilités.

> *Question 6.* Peut-on utiliser ce modèle statistique pour mieux appréhender les variations observées dans les ICO ?

Aperçu du contenu de la thèse

Après avoir décrit plus précisément le fonctionnement des ICO et présenté l'état de l'art sur l'apprentissage et les statistiques géométriques, les contributions de cette thèse sont organisées en deux parties.

Dans la première partie, nous décrivons les outils théoriques que nous apportons. L'outil principal est une distribution de probabilité qui prend en compte la géométrie des données, appelée la *gaussienne enveloppée*. Cette distribution sera étudiée théoriquement, et le problème de l'estimation de ses paramètres sera résolu. Grâce à cette gaussienne enveloppée, nous sommes capables de construire un cadre probabiliste regroupant de nombreux outils d'apprentissage géométrique déjà existants, tels que des classifieurs, des outils de détection d'observations aberrantes ou des noyaux. Nous sommes également capables de proposer de nouvelles extensions de ces outils en nous basant sur la gaussienne enveloppée. Enfin, nous verrons que nous pouvons réécrire des outils de flots génératifs déjà existants grâce à cette gaussienne enveloppée, tout en proposant un nouvel outil. Cette première partie répondra aux questions 4 et 5.

Dans la deuxième partie, nous nous focaliserons sur l'application aux ICO. Premièrement, nous étudierons une application particulière, qui est la détection de réveil en pleine anesthésie générale. Nous proposerons un nouveau classifieur basé sur la gaussienne enveloppée et nous

montrerons qu'il permet de mieux détecter les différents états lors d'une anesthésie générale. Cela répondra à la dernière question 6. Puis, nous nous éloignerons des problématiques données ci-dessus et, dans le dernier chapitre, nous proposerons différents outils d'interprétabilité des outils riemanniens utilisés en ICO afin de mieux comprendre comment ces outils prennent leurs décisions.

D.2 État de l'art

Dans l'introduction de cette thèse, nous avons problématisé ce travail autour de trois grands thèmes : les modèles d'apprentissage statistique et automatique, la variété $\mathcal{P}_d$ des matrices SDP et l'application aux ICO. Ce premier chapitre est consacré à l'état de l'art de ces trois thèmes.

D.2.1 Interfaces Cerveau-Ordinateur

L'application principale de cette thèse concerne les *Interfaces Cerveau-Ordinateur* (ICO), dont l'objectif est de traduire les signaux cérébraux en commandes concrètes. Dans cet état de l'art, nous décrivons les différentes étapes d'une ICO : l'enregistrement des signaux cérébraux grâce aux *ÉlectroEncéphaloGrammes* (EEG), le prétraitement des EEG, l'extraction de caractéristiques puis la classification de ces caractéristiques. Chaque étape est détaillée, ainsi que les techniques qui lui sont associées. Différentes applications des ICO sont ensuite décrites : le contrôle de prothèses ou de claviers virtuels, la détection d'un état de concentration, ou les applications médicales, de la surveillance d'anesthésies à l'utilisation en rééducation après un accident vasculaire cérébral. Les défis des ICO sont également évoqués, avec une attention particulière portée aux différents types de variabilités affectant les ICO, dont la modélisation a été la motivation initiale de ce travail.

Enfin, la caractéristique principale qui sera utilisée dans ce travail est la matrice de covariance, estimée à partir d'un EEG. Nous motivons donc ce choix, en montrant que cette matrice de covariance s'appuie sur des caractéristiques déjà bien utilisées en ICO tout en apportant de l'information supplémentaire. Nous motivons finalement l'utilisation d'une certaine géométrie pour ces matrices de covariance.

D.2.2 Contexte mathématique

Nous décrivons ensuite les outils mathématiques nécessaires à l'étude des matrices SDP. Nous commençons par rappeler des propriétés matricielles de ces dernières, puis justifions pourquoi la géométrie euclidienne n'est pas adaptée à la manipulation de ces matrices. Lorsque nous utilisons la géométrie euclidienne, deux effets négatifs principaux apparaissent : l'impossibilité d'extrapoler sans sortir de $\mathcal{P}_d$ et l'effet de grossissement, notamment évoqué par Arsigny et al. (2006a). Heureusement, nous pouvons utiliser le fait que $\mathcal{P}_d$ soit une variété et ainsi définir une géométrie riemannienne dessus.

Plusieurs géométries ont été définies, et si, dans cette thèse, nous nous focalisons sur la géométrie affine-invariante (Pennec 2020), nous décrivons dans ce chapitre plusieurs autres géométries, telles que la géométrie log-euclidienne (Arsigny et al. 2006a ; Arsigny et al. 2006b), la géométrie de Cholesky (Z. Wang et al. 2004), celle de log-Cholesky (Z. Lin 2019) et celle de Bures-Wasserstein (Bhatia et al. 2019 ; Malagò et al. 2018). Nous comparons également numériquement les différentes géométries.

D.2.3 État de l'art en statistiques géométriques

Enfin, nous donnons un large panorama des différentes méthodes existantes pour apprendre sur une variété. Nous nous intéressons à des outils d'apprentissage supervisé, aux méthodes à

Propriété	Gaussienne isotrope	Gaussienne à entropie maximale	**Gaussienne enveloppée**
Définie sur $\mathcal{P}_d$	✓	✓	✓
Anisotrope	✗	✓	✓
Échantillonnage simple	✗	✗	✓
Densité explicite	✗	✗	✓
Estimation des paramètres	∼*	✗	✓

* Un estimateur du maximum de vraisemblance (EMV) est connu pour le paramètre d'échelle, mais son calcul est coûteux en pratique en termes de temps de calcul.

Table D.1. Comparaison des principales distributions gaussiennes riemanniennes avec la gaussienne enveloppée proposée.

noyaux, aux outils de réduction de dimension, aux distributions de probabilité définies sur une variété ainsi qu'aux outils d'apprentissage profond. Pour chaque catégorie, nous décrivons les différentes méthodes connues, en insistant sur leurs points communs et leurs différences.

Dans la dernière partie, nous classons les différents outils en deux catégories : les outils *intrinsèques*, c'est-à-dire des outils directement définis sur la variété, et les outils *extrinsèques*, qui reposent sur un espace tangent pour être définis. Ces deux catégories seront utiles dans les chapitres de contributions qui suivent.

D.3 La gaussienne enveloppée

La loi normale, aussi connue sous le nom de distribution gaussienne, est omniprésente en probabilités, en statistiques et en apprentissage automatique pour modéliser des données. Une des raisons de cette omniprésence est le *Théorème Central Limite* (Wasserman 2004, TCL, voir Théorème 5.12 de), qui énonce qu'une somme de nombreuses variables aléatoires indépendantes suit approximativement une loi normale. Ainsi, de nombreux phénomènes physiques peuvent être vus comme une somme d'événements indépendants et donc suivre une loi normale. De nombreuses autres propriétés de la loi normale, telles que ses moments finis, sa stabilité par transformation linéaire et sa densité explicite, peuvent être très utiles. Lorsque l'on utilise cette loi normale pour modéliser des données, on fait une hypothèse importante : que les données vivent dans un espace euclidien plat. Nous avons vu que cela n'était pas le cas pour les matrices SDP, qui vivent plutôt sur une variété riemannienne courbe. Ainsi, l'objectif principal de ce premier chapitre est de **définir et étudier une gaussienne multivariée qui prenne en compte la géométrie des matrices SDP.**

Plus précisément, nous voulons que notre distribution

- soit bien définie sur la variété $\mathcal{P}_d$ des matrices SDP,
- soit anisotrope,
- soit facilement échantillonnable,
- ait une densité explicite,
- ait des paramètres qui puissent être estimés.

Ainsi, nous voulons une distribution qui soit à la fois bien définie et étudiée théoriquement, mais nous voulons également qu'elle soit utile en pratique. Nous ne sommes bien sûr pas les premiers à définir et à étudier des distributions sur les variétés riemanniennes, et nous pouvons principalement nous comparer à deux distributions : la gaussienne isotrope définie par Said, Bombrun et al. (2017) ainsi que la gaussienne à entropie maximale définie par Pennec (2006). Dans le tableau D.1, nous montrons que ces deux gaussiennes préexistantes ne satisfont pas toutes les propriétés que l'on souhaite pour notre distribution. C'est pourquoi nous introduisons une nouvelle version d'une gaussienne sur la variété $\mathcal{P}_d$.

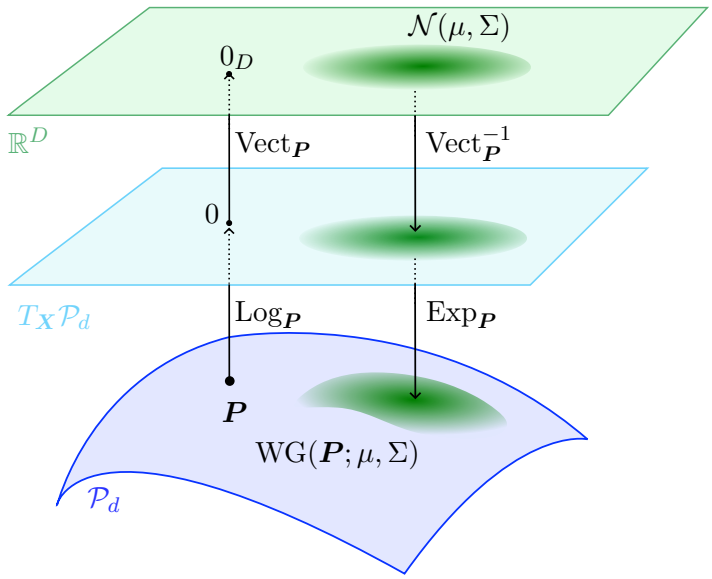


Figure D.2. Illustration de la gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ sur la variété des matrices symétriques définies positives $\mathcal{P}_d$. La gaussienne euclidienne $\mathcal{N}(\mu, \Sigma)$ est transportée de $\mathbb{R}^D$ vers l'espace tangent $T_{\boldsymbol{P}}\mathcal{P}_d$ au moyen de l'application $\mathrm{Vect}_{\boldsymbol{P}}^{-1}$. L'application exponentielle riemannienne $\mathrm{Exp}_{\boldsymbol{P}}$ permet ensuite de passer de $T_{\boldsymbol{P}}\mathcal{P}_d$ à la variété $\mathcal{P}_d$.

D.3.1 La gaussienne enveloppée

La distribution gaussienne enveloppée sur la variété des matrices définies positives $\mathcal{P}_d$ est définie en transportant une distribution gaussienne euclidienne classique depuis l'espace tangent en un point $\boldsymbol{P} \in \mathcal{P}_d$ vers la variété, au moyen de l'application exponentielle. Plus formellement, nous pouvons définir la distribution gaussienne enveloppée de la manière suivante :

Definition D.1 (Distribution gaussienne enveloppée)**.** Soient $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ et Σ une matrice définie positive de taille $D \times D$. Une variable aléatoire $\boldsymbol{X}$ à valeurs dans $\mathcal{P}_d$ suit une distribution gaussienne enveloppée, notée $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$, si

$$\boldsymbol{X} = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t)), \quad t \sim \mathcal{N}(\mu, \Sigma).$$

La gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ est donc la mesure image (voir la section 3.6 de Bogachev 2007) de la distribution gaussienne $\mathcal{N}(\mu, \Sigma)$ par l'application $\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1} : \mathbb{R}^D \to \mathcal{P}_d$:

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \left(\mathrm{Exp}_{\boldsymbol{P}} \circ \mathrm{Vect}_{\boldsymbol{P}}^{-1}\right) \# \mathcal{N}(\mu, \Sigma), \tag{D.1}$$

où $\#$ désigne l'opérateur de mesure image. La définition de la gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ fournit une méthode simple pour échantillonner cette distribution : il suffit de tirer un échantillon t suivant la distribution gaussienne $\mathcal{N}(\mu, \Sigma)$, puis de calculer $X = \mathrm{Exp}_{\boldsymbol{P}}(\mathrm{Vect}_{\boldsymbol{P}}^{-1}(t))$. Sur la variété des matrices définies positives $\mathcal{P}_d$ munie de la métrique affine-invariante, les applications $\mathrm{Exp}_{\boldsymbol{P}}$ et $\mathrm{Vect}_{\boldsymbol{P}}^{-1}$ admettent des expressions explicites ; il est donc possible de générer efficacement des échantillons suivant la distribution gaussienne enveloppée.

Telle que définie ci-dessus (définition D.1), la gaussienne enveloppée dépend de trois paramètres : $\boldsymbol{P} \in \mathcal{P}_d$, $\mu \in \mathbb{R}^D$ et $\Sigma \in \mathcal{P}_D$. Le paramètre $\boldsymbol{P}$ joue le rôle d'un point d'ancrage sur la variété $\mathcal{P}_d$ et définit l'espace tangent $T_{\boldsymbol{P}}\mathcal{P}_d$ à partir duquel la distribution gaussienne est enveloppée. Nous désignerons ce paramètre $\boldsymbol{P}$ par le terme *point d'ancrage de la gaussienne enveloppée*. Les deux autres paramètres correspondent aux paramètres classiques d'une distribution gaussienne multivariée sur $\mathbb{R}^D$, puisque l'espace tangent $T_{\boldsymbol{P}}\mathcal{P}_d$ est isométrique à $\mathbb{R}^D$ au moyen de l'application de vectorisation $\mathrm{Vect}_{\boldsymbol{P}}$: μ désigne l'espérance de la gaussienne sur $\mathbb{R}^D$ et $\mathrm{Vect}_{\boldsymbol{P}}^{-1}(\mu)$ modélise l'écart par rapport à l'origine de l'espace tangent. On peut in-

terpréter l'origine de $T_{\boldsymbol{P}}\mathcal{P}_d$ comme le point autour duquel l'espace tangent fournit la meilleure approximation linéaire locale de la variété $\mathcal{P}_d$ au voisinage de $\boldsymbol{P}$. Enfin, Σ désigne la matrice de dispersion de la gaussienne[1].

La figure D.2 illustre la gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ sur la variété $\mathcal{P}_d$. Il convient de noter que dans d'autres travaux, notamment ceux de Galaz-Garcia et al. (2022), Nagano et al. (2019) et Troshin et Niculae (2023), les auteurs se restreignent à des gaussiennes enveloppées centrées dans l'espace tangent $T_{\boldsymbol{P}}\mathcal{P}_d$, imposant ainsi $\mu = 0$. Dans ce travail, nous autorisons $\mu \neq 0$, ce qui confère au modèle une expressivité et une flexibilité accrues. En effet, sur une variété non plate, s'éloigner de l'origine de $T_{\boldsymbol{P}}\mathcal{P}_d$, puis se projeter à nouveau sur $\mathcal{P}_d$ au moyen de l'application exponentielle, ne revient pas nécessairement à modifier le point d'ancrage $\boldsymbol{P}$, mais peut conduire à définir une nouvelle distribution sur $\mathcal{P}_d$.

D.3.2 Propriétés de la gaussienne enveloppée

Après avoir défini la gaussienne enveloppée, nous sommes capables de donner plusieurs de ses propriétés. Nous listons et expliquons brièvement ici les différentes propriétés démontrées dans cette thèse.

- Grâce à la définition de la gaussienne enveloppée *via* une mesure image, nous pouvons calculer explicitement la **densité** de la gaussienne enveloppée. Cette densité dépend de la densité de la gaussienne multivariée dans l'espace tangent et de la jacobienne de l'application exponentielle riemannienne. Cette jacobienne permet de prendre en compte la déformation induite par le changement de variable entre l'espace tangent et la variété $\mathcal{P}_d$ elle-même. Dans le cadre de la variété $\mathcal{P}_d$, nous pouvons calculer explicitement cette jacobienne, et donc la densité complète.
- Nous donnons également une proposition précisant les **transformations** permettant de passer d'une gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma)$ à une gaussienne enveloppée *centrée réduite* $\mathrm{WG}(I_d; 0_D, I_D)$.
- Nous donnons ensuite quelques propositions portant sur certains des **moments** de la gaussienne enveloppée.
- Enfin, nous donnons une version enveloppée du Théorème Central Limite (TCL), en utilisant le *produit logarithmique* tel que défini par Arsigny et al. (2006a). Ce TCL enveloppé montre qu'un produit logarithmique de variables aléatoires indépendantes sur $\mathcal{P}_d$ suit une gaussienne enveloppée.

D.3.3 Une relation d'équivalence entre gaussiennes enveloppées

Une propriété importante d'un modèle statistique est son identifiabilité. On dit qu'un modèle statistique $\{\mathbb{P}_\theta, \theta \in \Theta\}$ est *identifiable* si $\theta \mapsto \mathbb{P}_\theta$ est injective, c'est-à-dire si $\theta_1 \neq \theta_2 \to \mathbb{P}_{\theta_1} \neq \mathbb{P}_{\theta_2}$. Pour la gaussienne enveloppée, nous avons montré que ce n'est pas le cas. En effet, nous avons le théorème suivant :

Theorem D.2 (Gaussiennes enveloppées équivalentes)**.** *Soient $\boldsymbol{P}, \boldsymbol{Q} \in \mathcal{P}_d$, $\mu, \nu \in \mathbb{R}^D$ et $\Sigma, \Gamma \in \mathcal{P}_D$. Alors, on a l'équivalence suivante :*

$$\mathrm{WG}(\boldsymbol{P}; \mu, \Sigma) = \mathrm{WG}(\boldsymbol{Q}; \nu, \Gamma) \iff \exists t \in \mathbb{R},\ \boldsymbol{Q} = e^t \boldsymbol{P}, \nu = \mu - t\ \mathrm{Vect}_{I_d}(I_d), \Sigma = \Gamma.$$

Ainsi, une certaine transformation des paramètres de la gaussienne enveloppée la laisse inchangée. La gaussienne enveloppée n'est donc pas identifiable. Pour régler ce problème, nous définissons une relation d'équivalence entre les paramètres définissant une même gaussienne enveloppée et nous choisissons un représentant dans chaque classe d'équivalence. Ce défaut

[1] Nous employons le terme *matrice de dispersion*, plutôt que *matrice de covariance*, afin d'éviter toute confusion avec les matrices de covariance des signaux EEG (qui constituent les éléments de $\mathcal{P}_d$) dans les applications.

d'identifiabilité vient du fait que nous avons autorisé la moyenne μ dans l'espace tangent à être différente de 0. De plus, toutes les gaussiennes enveloppées ne peuvent pas être réduites à des gaussiennes enveloppées centrées dans l'espace tangent. Autrement dit, toutes les classes d'équivalence ne contiennent pas un élément avec $\mu = 0$.

D.3.4 Estimation des paramètres

Un problème central en statistique consiste à estimer les paramètres d'un modèle statistique à partir d'un échantillon fini. Soient $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ un ensemble de N points de la variété $\mathcal{P}_d$, supposés indépendants et identiquement distribués selon une gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}^\star; \mu^\star, \Sigma^\star)$, dont les paramètres inconnus sont regroupés dans $\theta^\star = (\boldsymbol{P}^\star, \mu^\star, \Sigma^\star)$. Afin d'estimer $\theta^\star$ à partir de l'échantillon $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N)$, nous considérons l'estimateur du maximum de vraisemblance (EMV) (voir la section 9.3 de Wasserman 2004), qui consiste à maximiser la fonction de vraisemblance :

$$\forall (\boldsymbol{P}, \mu, \Sigma) \in \mathcal{P}_d \times \mathbb{R}^D \times \mathcal{P}_D, \quad \mathcal{L}_N(\boldsymbol{P}, \mu, \Sigma) = \prod_{i=1}^{N} f_{\boldsymbol{P};\mu,\Sigma}(\boldsymbol{X}_i), \tag{D.2}$$

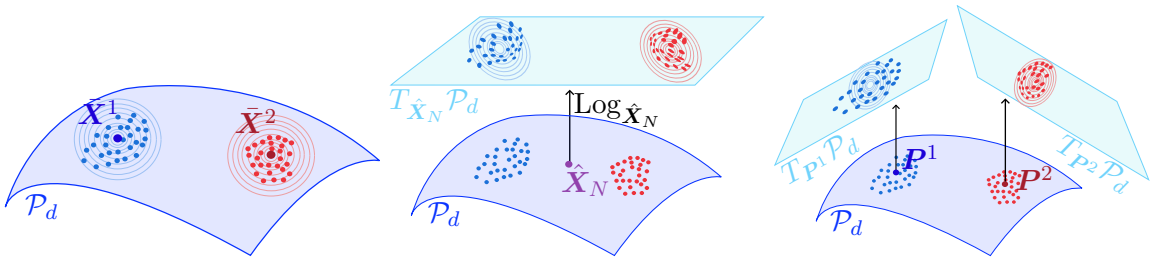
où $f_{\boldsymbol{P};\mu,\Sigma}$ désigne la densité de la gaussienne enveloppée.

Dans le cadre euclidien classique de la loi normale, on dispose d'expressions explicites des EMV de la gaussienne. Pour la gaussienne enveloppée, c'est le cas pour les EMV $\hat{\mu}$ et $\hat{\Sigma}$ des paramètres dans l'espace tangent (μ, Σ), mais ces derniers dépendent du vrai point d'ancrage $\boldsymbol{P}^\star$, qui est inconnu. Nous n'avons pas d'expression explicite pour l'EMV $\hat{\boldsymbol{P}}$ de $\boldsymbol{P}$ et nous ne pensons pas qu'il en existe une. Pour résoudre ce problème, nous utilisons un algorithme d'optimisation riemannien, tel que présenté dans Boumal (2023), permettant de résoudre numériquement le problème d'optimisation de l'équation D.2.

Nous avons conduit des expériences synthétiques pour vérifier la performance de cet estimateur. Pour cela, nous avons étudié l'évolution de l'erreur d'estimation en fonction du nombre d'échantillons et de la dimension des matrices. Nous avons montré que plus le nombre d'échantillons est grand, plus l'erreur d'estimation est petite, et que la dimension a surtout une influence sur l'erreur d'estimation de la matrice de dispersion Σ.

D.3.5 Extension à d'autres variétés

Dans ce premier chapitre de contributions, nous avons défini et étudié la gaussienne enveloppée sur la variété $\mathcal{P}_d$ des matrices SDP, mais sa définition ne dépend pas des spécificités de cette variété. Ainsi, dans la dernière partie de ce chapitre, nous étendons la définition de la gaussienne enveloppée à une classe plus large de variétés : les *variétés d'Hadamard*. Une variété d'Hadamard est une variété riemannienne complète, simplement connexe et de courbure sectionnelle négative ou nulle partout. Des exemples de variétés d'Hadamard sont l'espace euclidien $\mathbb{R}^d$, l'espace des matrices SDP $\mathcal{P}_d$ et l'espace hyperbolique $\mathbb{H}^d$. Nous expliquons alors comment, grâce au théorème de Cartan-Hadamard (voir le théorème 12.8 de J. M. Lee 2018), nous pouvons définir correctement les gaussiennes enveloppées et étendre certaines de leurs propriétés aux variétés d'Hadamard. Nous nous intéressons ensuite à la question de l'identifiabilité sur différentes variétés d'Hadamard, en montrant que la gaussienne enveloppée n'est identifiable ni sur $\mathbb{R}^d$, ni sur $\mathcal{P}_d$, alors qu'elle l'est sur $\mathbb{H}^d$. Nous relions alors cette question à la structure de l'espace, notamment *via* la *décomposition de de Rham* (Sakai 1996, voir théorème 6.11 de), et proposons une conjecture pour l'identifiabilité de la gaussienne enveloppée sur une variété d'Hadamard générale.



(a) MDM. **(b)** TS-LDA. **(c)** Ho-WDA.

Figure D.3. Illustration des différents classifieurs riemanniens considérés dans cette étude sur un problème à deux classes.

D.4 Construire un cadre probabiliste pour l'apprentissage automatique sur $\mathcal{P}_d$

La distribution gaussienne est très souvent utilisée en apprentissage automatique pour modéliser les données. De nombreux algorithmes sont ensuite construits en faisant cette hypothèse. Dans ce chapitre, nous montrons comment, grâce à la gaussienne isotrope de Said, Bombrun et al. (2017) et à la gaussienne enveloppée étudiée à la section D.3, nous sommes capables de réécrire de nombreux algorithmes d'apprentissage automatique développés sur $\mathcal{P}_d$ dans un cadre probabiliste commun. De plus, nous proposons des extensions de méthodes déjà existantes grâce à la gaussienne enveloppée. Notre cadre probabiliste englobe plusieurs types d'outils : des classifieurs, des outils de détection d'observations aberrantes et des noyaux.

D.4.1 Les classifieurs

Pour les classifieurs, nous nous plaçons dans le cadre d'un *classifieur du maximum de vraisemblance*.

Definition D.3 (Classifieur du maximum de vraisemblance)**.** Étant donné K classes, chacune étant modélisée par une distribution α_k, un *classifieur du maximum de vraisemblance* assigne un nouvel échantillon $\boldsymbol{Z}$ à la classe k qui maximise la vraisemblance de $\boldsymbol{Z}$ sous la distribution α_k :

$$\hat{y} = \arg\max_k f_k(\boldsymbol{Z}),$$

où f_k désigne la densité de probabilité de α_k.

En fonction du choix des distributions α_k des différentes classes, nous sommes capables de réécrire différents classifieurs déjà connus sur $\mathcal{P}_d$. En particulier, nous nous focalisons sur deux classifieurs de l'état de l'art : le *Minimum Distance to the Mean* (MDM) proposé par Barachant, Bonnet et al. (2010b) et le *Tangent Space LDA* (TS-LDA) proposé par Barachant, Bonnet et al. (2011).

En particulier, nous montrons que le MDM modélise chaque classe k par une gaussienne isotrope centrée en sa propre moyenne de Fréchet $\bar{\boldsymbol{X}}^k$. Le TS-LDA, quant à lui, projette l'ensemble des classes dans un unique espace tangent commun $T_{\hat{\boldsymbol{X}}_N}\mathcal{P}_d$, au point $\hat{\boldsymbol{X}}_N$ correspondant à la moyenne de Fréchet estimée à partir de l'ensemble des données d'entraînement, puis s'appuie sur une LDA classique (Hastie et al. 2009, voir section 4.3 de). Ainsi, il est possible de réécrire le TS-LDA en un classifieur du maximum de vraisemblance en modélisant chaque classe par une gaussienne enveloppée ancrée en la moyenne de Fréchet globale $\hat{\boldsymbol{X}}_N$ et avec des paramètres spécifiques dans l'espace tangent pour chaque classe.

Nous proposons également un nouveau classifieur qui modélise chaque classe grâce à une gaussienne enveloppée ancrée au niveau de chaque classe. Nous proposons deux versions : l'une dans laquelle la matrice de dispersion Σ est partagée entre les classes et l'autre où chaque

classe a sa propre matrice de dispersion (à la manière de la LDA/QDA). Nous nommons ces classifieurs *Wrapped Discriminant Analysis* (WDA).

Les différents classifieurs sont illustrés à la figure D.3 et nous avons comparé quantitativement leurs performances sur différents jeux de données réels. Nous avons montré que, avec suffisamment de données d'entraînement, notre WDA surpasse les autres classifieurs (MDM et TS-LDA).

D.4.2 Outils de détection d'observations aberrantes

Un deuxième type d'outils que nous avons dans notre cadre probabiliste est un outil classique de détection d'observations aberrantes en ICO riemannienne : la patate riemannienne développée par Barachant, Andreev et al. (2013). Nous la réinterprétons dans notre cadre probabiliste en montrant que sa règle de décision est un seuil sur la vraisemblance sous une gaussienne isotrope. Nous proposons également une extension à l'aide des gaussiennes enveloppées.

D.4.3 Noyaux

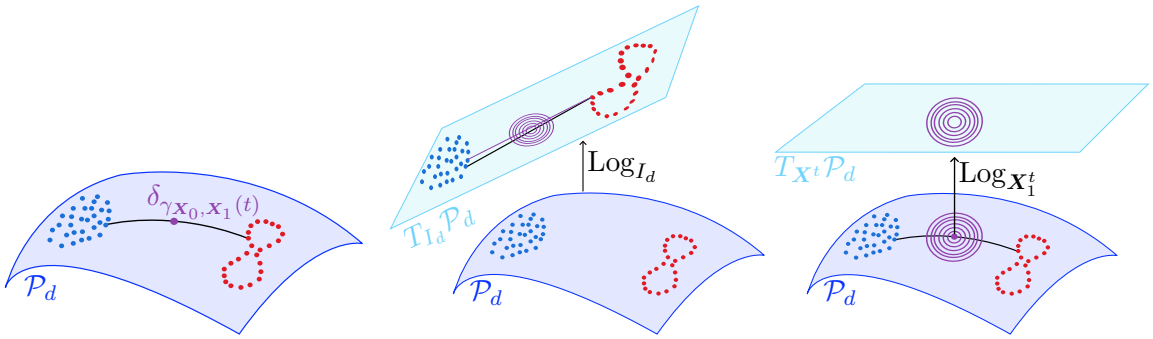
Le dernier type d'outils étudié dans ces chapitres est celui des noyaux. Après avoir rappelé un résultat crucial de Feragen et al. (2015), qui lie les noyaux gaussiens à la géométrie de l'espace ambiant, nous commençons par étudier les noyaux déjà existants dans la littérature. Ces deux noyaux sont le noyau AIRM, notamment étudié par Wagner vom Berg (2025), et le noyau log-euclidien, étudié notamment par Jayasumana et al. (2013). Nous proposons également une extension de ce noyau log-euclidien grâce à la gaussienne enveloppée. Notre nouveau noyau, que nous nommons *Noyau Gaussien Enveloppé*, prend notamment en compte la déformation induite par l'envoi des points de la variété vers l'espace tangent.

Nous comparons ensuite ces différents noyaux sur plusieurs jeux de données réels. Nos résultats ne montrent pas de supériorité nette entre les différents noyaux, mais montrent l'importance de bien choisir le point d'ancrage pour le noyau log-euclidien et pour le noyau gaussien enveloppé proposé.

D.5 Flots génératifs enveloppés

Un autre avantage de la gaussienne euclidienne classique est que de nombreux algorithmes d'échantillonnage ont été étudiés (Box et M. E. Muller 1958 ; Marsaglia et Tsang 2000). Ainsi, des méthodes très efficaces existent. Cela se transfère à la gaussienne enveloppée définie et étudiée dans cette thèse. Cette facilité d'échantillonnage de la gaussienne est à la base de nombreuses méthodes génératives, telles que les modèles de diffusion (Ho et al. 2020 ; Yang Song et al. 2021) ou les flots génératifs (Lipman, R. T. Q. Chen et al. 2023 ; X. Liu et al. 2023 ; Michael S Albergo et Vanden-Eijnden 2022), qui seront au cœur de ce chapitre. L'objectif d'un modèle génératif est de générer de nouveaux points suivant une distribution de probabilité inconnue, dont on ne connaît qu'un certain nombre d'échantillons. Par exemple, on dispose d'un jeu de données composé d'images, mais on ne connaît pas la distribution sous-jacente de ces images. Pour générer une nouvelle image réaliste, on peut utiliser une méthode générative.

Dans ce chapitre, nous nous focalisons sur les *flots génératifs* (Lipman, R. T. Q. Chen et al. 2023 ; X. Liu et al. 2023 ; Michael S Albergo et Vanden-Eijnden 2022). L'idée derrière ces flots génératifs est de partir d'une distribution source p simple à échantillonner (typiquement une gaussienne dans le cas euclidien) et d'apprendre une transformation ϕ qui transforme la distribution source p en la distribution cible q (dont on rappelle que l'on n'a accès qu'à un nombre fini d'échantillons). Ici, la transformation entre p et q est modélisée par un flot dépendant du temps, défini comme la solution d'une équation différentielle ordinaire (EDO). Plutôt que d'apprendre directement cette transformation, l'objectif est d'estimer le



(a) RiemannianCFM. **(b)** DiffeoCFM. **(c)** WrappedCFM.

Figure D.4. Illustration des différentes méthodes de CFM sur la variété $\mathcal{P}_d$: le RiemannianCFM, qui utilise l'interpolation géodésique entre $\boldsymbol{X}_0$ et $\boldsymbol{X}_1$; le DiffeoCFM, qui repose sur une application dans l'espace tangent en l'identité I_d, suivie de l'application du CFM euclidien ; et enfin, le WrappedCFM proposé, qui utilise une gaussienne enveloppée dont le point d'ancrage se déplace le long de la géodésique reliant I_d à $\boldsymbol{X}_1$.

champ de vecteurs dépendant du temps qui la gouverne, généralement à l'aide d'un réseau de neurones. La génération de nouveaux échantillons consiste alors à intégrer numériquement l'EDO associée à ce champ de vecteurs.

L'apprentissage de ce champ de vecteurs à partir de simples échantillons de la distribution cible constitue une difficulté centrale. Les *flots génératifs conditionnels* (Lipman, R. T. Q. Chen et al. 2023) y répondent en introduisant un apprentissage conditionnel du champ de vecteurs, tout en permettant de retrouver le champ global recherché. Ces méthodes ont notamment obtenu des résultats de pointe en génération d'images (Esser et al. 2024), de vidéos (A. Polyak et al. 2024), de parole (Le et al. 2023) et de signaux audio (Vyas et al. 2023).

D.5.1 Flots génératifs riemanniens et WrappedCFM

Ces méthodes de flots génératifs ont également été étendues à des variétés riemanniennes à travers deux méthodes : *RiemannianCFM*, présentée dans R. T. Q. Chen et Lipman (2024), et *DiffeoCFM*, issue de Collas, Ju et al. (2026). Après avoir détaillé le fonctionnement des flots génératifs euclidiens, nous présentons les deux adaptations riemanniennes mentionnées ci-dessus. Le RiemannianCFM se base sur l'interpolation géodésique et son chemin de probabilités conditionnelles est donc composé de mesures de Dirac suivant la géodésique. Pour DiffeoCFM, la méthode envoie la distribution cible dans un espace tangent bien choisi (typiquement en l'identité I_d), s'appuie sur un flot génératif euclidien classique dans cet espace tangent, puis renvoie les points échantillonnés sur la variété.

Dans ce chapitre, nous proposons un cadre global permettant de rassembler les deux méthodes décrites ci-dessus, RiemannianCFM et DiffeoCFM. Pour cela, nous introduisons un chemin de probabilités conditionnelles général $(p_t(\cdot \mid \boldsymbol{Z}))_{(t)}$, en utilisant la gaussienne enveloppée :

$$p_t(\cdot \mid \boldsymbol{Z}) = \mathrm{WG}(\bar{\boldsymbol{X}}_t(\boldsymbol{Z}); \mu_t(\boldsymbol{Z}), \sigma_t(\boldsymbol{Z})^2 I_D),$$

où les fonctions $\bar{\boldsymbol{X}} \colon [0,1] \times \mathcal{Z} \to \mathcal{P}_d$, $\mu \colon [0,1] \times \mathcal{Z} \to \mathbb{R}^D$ et $\sigma \colon [0,1] \times \mathcal{Z} \to \mathbb{R}_{\geqslant 0}$ paramétrisent le chemin. À partir de ce chemin de probabilités conditionnelles, nous calculons le flot permettant de le générer ainsi que le champ de vecteurs associé à ce flot. Nous montrons que ce champ de vecteurs dépend de la valeur de $\sigma_t(\boldsymbol{Z})$, en distinguant le cas où la distribution se résume à un Dirac ($\sigma_t(\boldsymbol{Z}) = 0$) du cas où la distribution est à support infini ($\sigma_t(\boldsymbol{Z}) > 0$).

Ce cadre général nous permet, en fonction du choix de la paramétrisation du chemin de probabilités conditionnelles (c'est-à-dire du choix de $\bar{\boldsymbol{X}}, \mu$ et σ), de retrouver les méthodes

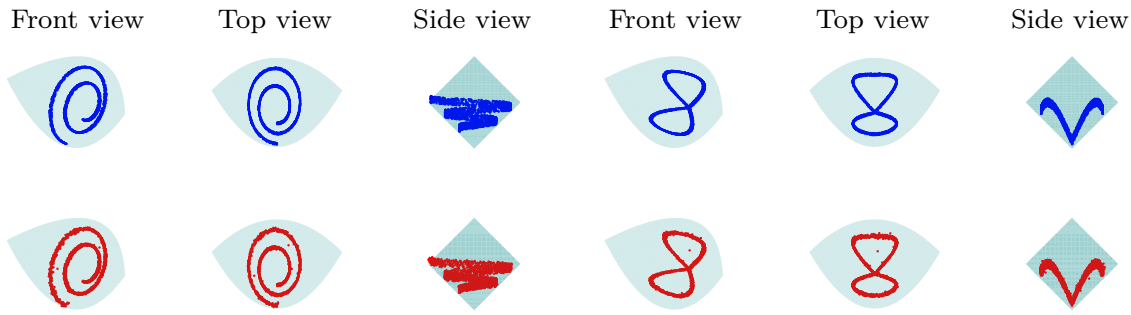


Figure D.5. Deux exemples illustrant les résultats obtenus avec le modèle WrappedCFM proposé une fois le flot appris. La distribution cible réelle apparaît sur la ligne du haut et celle apprise sur la ligne du bas. Dans ces exemples, chaque point représente une matrice SDP de taille 2×2, qui peut ainsi être représentée en 3D. On constate que, d'un point de vue qualitatif, le modèle WrappedCFM proposé est capable d'apprendre les distributions correctes.

RiemannianCFM et DiffeoCFM. De plus, en proposant la nouvelle paramétrisation suivante :

$$\bar{\boldsymbol{X}}_t(\boldsymbol{X}_1) = \boldsymbol{X}_1^t,\ \mu_t(\boldsymbol{X}_1) = 0_D,\ \text{et}\ \sigma_t(\boldsymbol{X}_1) = \sigma_{\text{init}} - (\sigma_{\text{init}} - \sigma_{\min})\, t,$$

nous définissons une nouvelle méthode de flot génératif sur la variété $\mathcal{P}_d$, que nous nommons *WrappedCFM*. Nous instancions alors les flots et champs de vecteurs généraux dans le cadre de cette paramétrisation spécifique.

Le WrappedCFM se distingue du RiemannianCFM par l'utilisation d'une distribution à support infini, tandis que ce dernier repose sur des distributions de Dirac (il constitue ainsi un cas dégénéré de notre cadre, obtenu lorsque $\sigma_t(\boldsymbol{X}_1) = 0$). Par ailleurs, contrairement au DiffeoCFM, qui projette systématiquement les données dans un unique espace tangent $T_{I_d}\mathcal{P}_d$, le WrappedCFM adapte progressivement son espace tangent grâce au déplacement continu du point d'ancrage de la gaussienne enveloppée le long d'une géodésique. En effet, si $T_{I_d}\mathcal{P}_d$ constitue une bonne approximation locale de la variété autour de l'identité, cette approximation peut engendrer des déformations lorsque la distribution cible est très dispersée (Jaquier, Rozo et Asfour 2024). Le déplacement de l'ancrage permet ainsi au WrappedCFM de suivre le centre de masse des chemins de probabilité conditionnels et de mieux prendre en compte la géométrie de la variété au cours du processus de génération. Les différentes méthodes de flots génératifs riemanniens sont illustrées à la figure D.4.

D.5.2 Expériences numériques

Nous voulons ensuite comparer numériquement les différentes méthodes de flots génératifs riemanniens étudiées ici : le RiemannianCFM, le DiffeoCFM et le WrappedCFM que nous proposons. Pour cela, nous commençons par mener des expériences synthétiques, puis passons à des expériences sur des jeux de données réels.

Expériences synthétiques Nous commençons par mener deux expériences synthétiques. La première permet de démontrer que la méthode que nous proposons (WrappedCFM) est bien capable d'apprendre qualitativement une distribution cible. Pour cela, nous créons deux jeux de données synthétiques composés de matrices SDP de taille 2×2, qui peuvent donc être visualisées en trois dimensions. Nous illustrons ces jeux de données ainsi que le résultat de l'apprentissage de WrappedCFM à la figure D.5. La seconde expérience synthétique démontre la robustesse des différentes méthodes face à des matrices mal conditionnées. Pour cela, nous créons un jeu de données composé d'une mixture de gaussiennes enveloppées dont nous contrôlons le conditionnement des points d'ancrage. Nous montrons quantitativement que plus

le conditionnement des matrices est grand, moins DiffeoCFM est performant. Nous montrons également qu'une méthode de flot génératif euclidien classique est très peu performante.

Expériences sur des jeux de données réels Nous comparons ensuite nos différentes méthodes sur quatre jeux de données réels, issus d'applications différentes. Nous utilisons deux critères pour comparer les méthodes : premièrement, nous mesurons des métriques de qualité de la distribution (la densité et la couverture issues de Naeem et al. (2020) et la distance de Wasserstein). Deuxièmement, nous quantifions la qualité de la distribution générée dans une tâche de classification en suivant la méthode d'évaluation de Collas, Ju et al. (2026). Nous montrons que la méthode que nous proposons (WrappedCFM) se comporte de manière similaire au RiemannianCFM et est meilleure que le DiffeoCFM.

Ce travail est issu d'une collaboration avec une autre doctorante, Aimi Okabayashi, et son directeur de thèse, Nicolas Courty.

Transition : de la théorie aux applications

Dans les premiers chapitres de cette thèse, nous avons introduit l'outil central de ces travaux, la *gaussienne enveloppée*, et construit un cadre probabiliste qui englobe de nombreux outils d'apprentissage sur la variété des matrices SDP $\mathcal{P}_d$. Ce cadre englobe des classifieurs, des outils de détection d'observations aberrantes ou des méthodes de flots génératifs. Il permet à la fois d'englober des méthodes déjà existantes et de proposer de nouveaux outils basés sur la gaussienne enveloppée. Cela répond aux deux premières questions que nous avons posées en introduction de ce travail. Cependant, un aspect clé de la première question est **l'utilité** de ce nouveau modèle statistique. Cette utilité va notamment être démontrée dans le prochain chapitre, qui répond également à la troisième question, tournée vers les applications en Interfaces Cerveau-Ordinateur.

D.6 Classification à une classe pour la détection d'état d'anesthésie

L'objectif de ce chapitre est d'appliquer les outils développés à partir des gaussiennes enveloppées à un problème particulier d'ICO. Le problème qui va nous intéresser dans ce chapitre est celui de la *détection de réveil accidentel lors d'une anesthésie générale*. Ce problème survient lorsque la profondeur de l'anesthésie d'un patient n'est pas suffisante pour le garder endormi. Le patient peut alors se réveiller, ressentir des douleurs et garder, par la suite, des souvenirs d'événements liés à l'opération. Ces réveils accidentels surviennent dans environ 1 % des opérations à risque (Tasbihgou et al. 2018 ; JJ Pandit et Cook 2013) et construire un outil pour les détecter est donc crucial. Pour cela, nous allons utiliser une *stimulation du nerf médian* au poignet des patients. En stimulant ce nerf de manière indolore, tout en enregistrant en parallèle l'activité électrique du cerveau grâce à des EEG, il est possible de distinguer si le patient dort ou s'il est réveillé. L'objectif de ce chapitre est donc de construire un classifieur capable de distinguer les deux états à partir d'EEG enregistrés pendant des stimulations du nerf médian.

D.6.1 Description du jeu de données

Dans ce chapitre, nous travaillons sur un jeu de données collecté par une autre doctorante, Valérie Marissens Cueva, à l'hôpital Brugmann en Belgique. Ce jeu de données a été enregistré en deux temps : dans un premier temps, des EEG ont été enregistrés lors d'une première session préopératoire, alors que le patient était réveillé et conscient. Puis, dans un deuxième

temps, des EEG ont été enregistrés lors de l'opération, alors que le patient était sous anesthésie. Lors des deux sessions, le nerf médian du patient était stimulé.

D.6.2 Les classifieurs à une classe

Lors de la conception et de l'entraînement d'un classifieur, on a typiquement accès à toutes les classes dès l'entraînement. Dans cette application spécifique, ce n'est pas le cas : avant le début de l'anesthésie, nous n'avons accès qu'à la classe de l'état de réveil et non à celle correspondant à l'état anesthésié. Ainsi, nous ne pouvons entraîner notre classifieur que sur une seule des deux classes, celle de l'état de réveil. Nous allons donc utiliser des classifieurs à une classe (Perera et al. 2021). Dans cette étude, nous comparons deux classifieurs à une classe : le *One-Class MDM* et le *One-Class WG*.

One-Class MDM Ce classifieur a déjà été étudié par Marissens Cueva, Rimbert, Cebolla Alvarez et al. (2024) et repose sur le classifieur MDM déjà présenté. Étant donné la classe de l'état réveillé, la moyenne de Fréchet de cette classe est estimée lors de l'entraînement. Étant donné un nouveau point à classifier, sa distance à la moyenne de la classe réveillée est calculée et il est classifié comme appartenant à la classe réveillée s'il est suffisamment proche de la moyenne. Il s'agit d'une version hors ligne de la patate riemannienne de Barachant, Andreev et al. (2013).

One-Class WG Nous introduisons un nouveau classifieur à une classe, appelé *One-Class WG*, fondé sur la gaussienne enveloppée. À partir de N matrices symétriques définies positives (SDP) appartenant à la classe réveillée, le modèle ajuste une gaussienne enveloppée $\mathrm{WG}(\boldsymbol{P}; 0_D, \Sigma)$, caractérisée par un point d'ancrage $\boldsymbol{P}$, une moyenne nulle dans l'espace tangent et une matrice de dispersion Σ.

Bien que le modèle gaussien enveloppé général, autorisant une moyenne tangentielle non nulle, soit plus expressif, son estimation nécessite davantage de données. Nous choisissons donc de considérer une distribution centrée dans l'espace tangent ($\mu = 0$), ce qui permet également de résoudre le problème d'identifiabilité du modèle : la famille ainsi restreinte est identifiable.

En pratique, le point d'ancrage $\boldsymbol{P}$ est estimé par la moyenne de Fréchet des données d'entraînement, tandis que la matrice de dispersion Σ est obtenue par maximum de vraisemblance dans l'espace tangent associé. Pour classer une nouvelle matrice SDP $\boldsymbol{Z}$, on calcule sa log-vraisemblance sous la distribution estimée. L'échantillon est alors attribué à la classe réveillée si sa log-vraisemblance dépasse un seuil ε, et à la classe anesthésiée dans le cas contraire.

D.6.3 Les résultats

Nous comparons les performances du One-Class WG et du classifieur classique One-Class MDM sur l'ensemble des sujets. Compte tenu du fort déséquilibre entre les classes, la précision équilibrée (*balanced accuracy*) est retenue comme métrique d'évaluation. Le One-Class MDM obtient une précision équilibrée moyenne de $78{,}67\,\% \pm 15{,}93$, contre $83{,}88\,\% \pm 11{,}69$ pour le One-Class WG. Un test de Student apparié bilatéral confirme que cette amélioration est statistiquement significative. À l'échelle individuelle, le One-Class WG améliore sensiblement les performances pour 13 des 19 sujets.

Ces résultats sont obtenus avec seulement cinq électrodes (C3, C4, Cz, FC4 et FC3) et une matrice de dispersion diagonale pour le One-Class WG. Une étude d'ablation examine ensuite l'influence du nombre d'électrodes, de la structure de la matrice de dispersion et du choix du seuil de log-vraisemblance.

Dans cette configuration, le One-Class WG surpasse donc significativement le One-Class MDM. Ce résultat s'explique notamment par sa capacité à prendre en compte l'anisotropie

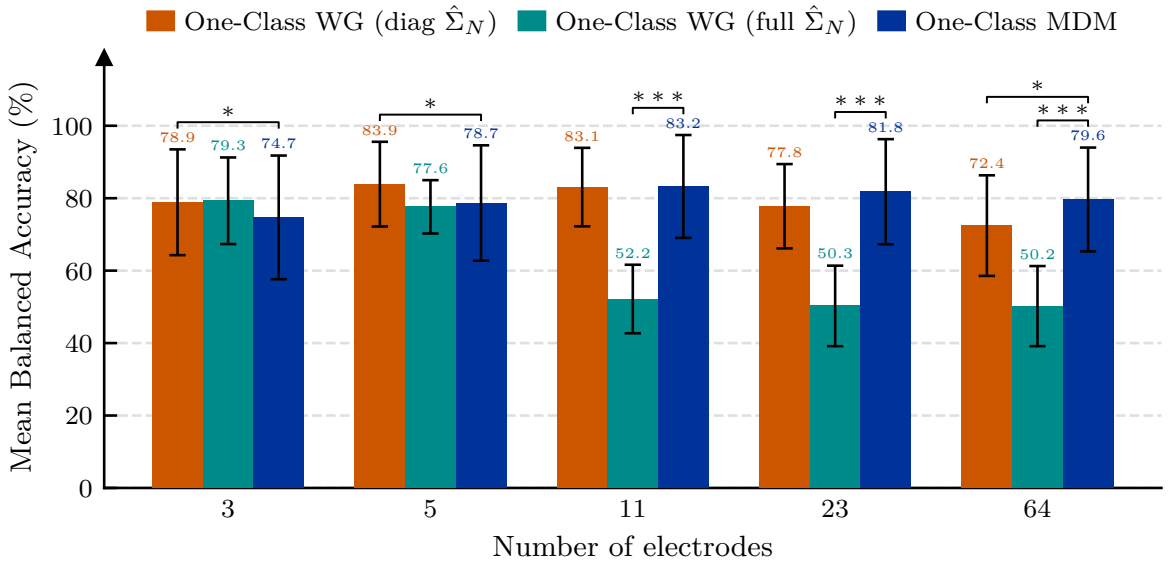


Figure D.6. Précision équilibrée moyenne des différents classifieurs à une classe selon différentes configurations d'électrodes. La significativité statistique a été évaluée à l'aide d'un test de Student apparié bilatéral (*$p < 0{,}05$, **$p < 0{,}01$, ***$p < 0{,}001$).

potentielle des données, contrairement au One-Class MDM, qui repose sur une hypothèse de dispersion isotrope. Le One-Class WG offre ainsi une modélisation plus expressive des données.

D.6.4 Étude d'ablation

Après avoir présenté le résultat dans une configuration particulière, nous avons conduit une étude d'ablation sur plusieurs paramètres : le nombre d'électrodes considérées, la structure de la matrice de dispersion Σ estimée et le seuil de classification pour le One-Class WG. Les résultats sont donnés à la figure D.6.

Nombre d'électrodes. L'influence du nombre d'électrodes est étudiée en considérant différentes configurations, de 3 à 64 électrodes. Les résultats montrent que les performances du One-Class WG diminuent lorsque le nombre d'électrodes augmente, probablement en raison du nombre limité d'essais disponibles pour estimer les paramètres du modèle. Les meilleures performances sont obtenues avec seulement 5 électrodes, avec une amélioration de la précision moyenne pouvant atteindre 5 points par rapport au One-Class MDM. Cette réduction du nombre d'électrodes présente également un intérêt pratique en contexte chirurgical, puisqu'elle simplifie et accélère la préparation du patient.

Structure de la matrice de dispersion. Deux structures sont comparées pour la matrice Σ : une matrice diagonale et une matrice pleine. La première réduit le nombre de paramètres à estimer et limite ainsi les erreurs d'estimation lorsque les données sont peu nombreuses. Elle donne généralement de meilleurs résultats à partir de 5 électrodes. En revanche, avec seulement 3 électrodes, la matrice pleine permet de mieux modéliser les dépendances entre les signaux et obtient les meilleures performances.

Seuil de décision. Enfin, plusieurs seuils sont évalués, définis à partir des 85^{e}, 90^{e}, 95^{e} et 99^{e} percentiles des log-vraisemblances des données d'entraînement. Leur influence sur les performances reste limitée, ce qui suggère une certaine robustesse du modèle à ce choix. Le

seuil correspondant au 95^{e} percentile fournit néanmoins les meilleurs résultats et est donc retenu pour la suite des expériences.

D.6.5 Discussion et analyse des résultats

Nous proposons également une discussion portant sur l'évolution des scores avec la profondeur de l'anesthésie, la robustesse en contexte chirurgical et l'intérêt de modéliser la dispersion des données.

Évolution des scores selon la profondeur de l'anesthésie. L'analyse des scores montre que les deux modèles réagissent à l'augmentation de la concentration de propofol, mais avec des performances variables selon les sujets. Le One-Class WG offre notamment une séparation plus cohérente pour certains sujets, potentiellement grâce à sa capacité à modéliser la dispersion des données, contrairement au MDM, qui repose uniquement sur la distance à un centroïde. Cette propriété pourrait être particulièrement utile en présence de phénomènes de *burst suppression*, associés à une forte variabilité des covariances sous anesthésie profonde.

Robustesse en environnement clinique. Les données acquises en conditions chirurgicales réelles présentent de nombreuses sources de variabilité, liées notamment au bruit, au positionnement des patients et à l'hétérogénéité des protocoles anesthésiques. Malgré ces contraintes, le One-Class WG surpasse le One-Class MDM, ce qui suggère que la modélisation de la dispersion de la classe réveillée améliore la robustesse du classifieur.

Isotropie et anisotropie de la dispersion. L'hypothèse avancée est que les meilleures performances du WG proviennent de sa capacité à modéliser la dispersion anisotrope des données. Pour l'étudier, la variance des coefficients diagonaux de $\hat{\Sigma}_N$ est comparée entre les groupes de sujets selon la méthode la plus performante. Elle est environ 16 fois plus élevée lorsque le WG surpasse le MDM ($3{,}021 \times 10^{-2}$ contre $1{,}891 \times 10^{-3}$), ce qui suggère que le WG est particulièrement avantageux lorsque les données présentent une dispersion anisotrope.

Ce travail est issu d'une collaboration avec une autre doctorante, Valérie Marissens Cueva, ses encadrants et les médecins qui l'ont aidée à collecter le jeu de données utilisé. Ce chapitre répond donc à la dernière question posée en introduction : la gaussienne enveloppée peut être utile dans le cadre d'une application réelle en ICO, ici pour la détection d'états d'anesthésie, notamment grâce à son anisotropie.

D.7 Interprétabilité d'outils riemanniens

Le dernier chapitre de cette thèse fait un pas de côté par rapport aux questions initialement posées dans l'introduction. En effet, ce chapitre est consacré à des outils d'interprétabilité pour les méthodes riemanniennes utilisées en ICO. L'objectif est de comprendre comment un modèle donné prend ses décisions. Cette question est très importante en apprentissage automatique et spécifiquement en ICO, où les différents utilisateurs ont des compétences très diverses. Notre objectif est de pouvoir expliquer une décision prise par un classifieur riemannien à un spécialiste des ICO qui n'est pas nécessairement formé à la géométrie riemannienne, à un médecin ou à l'utilisateur final d'une ICO.

De plus, lors de la construction d'une ICO, certains biais ou facteurs confondants peuvent être exploités par les algorithmes d'apprentissage pour améliorer leurs performances, sans pour autant s'appuyer sur une activité neuronale pertinente. Un classifieur cherchant uniquement à optimiser ses performances sur les données d'entraînement ne distingue pas les véritables signaux neurophysiologiques des artefacts, tels que les activités musculaires ou oculaires

(Fatourechi et al. 2007 ; Trocellier, N'Kaoua et al. 2024). Les outils d'interprétabilité permettent ainsi de détecter ces biais et de contribuer au développement d'ICO plus fiables.

Pour cela, nous avons développé deux types d'outils : un premier outil de visualisation de nuages de points composés de matrices SDP de grande dimension, et des outils plus classiques d'interprétabilité (Molnar 2025).

D.7.1 Le t-SNE riemannien

Le premier outil développé dans ce chapitre est une adaptation de l'algorithme du *t-SNE*, introduit par Maaten et Hinton (2008) dans le cadre euclidien, au cadre riemannien des matrices SDP. L'objectif de cet algorithme est de réduire la dimension des données afin de les visualiser, tout en conservant qualitativement les similarités présentes en grande dimension. Dans le cadre des matrices SDP, l'espace $\mathcal{P}_2$ des matrices SDP de taille 2×2 peut s'écrire comme suit :

$$\mathcal{P}_2 = \left\{ \begin{pmatrix} a & b \\ b & c \end{pmatrix} \mid a > 0, c > 0, ac > b^2 \right\}.$$

Ainsi, il est possible de représenter dans $\mathbb{R}^3$ une matrice SDP de taille 2×2 en utilisant les coordonnées (a, b, c). Ces points se situent à l'intérieur du cône d'équation $b = \pm\sqrt{ac}$, qui est représenté à la figure D.7.

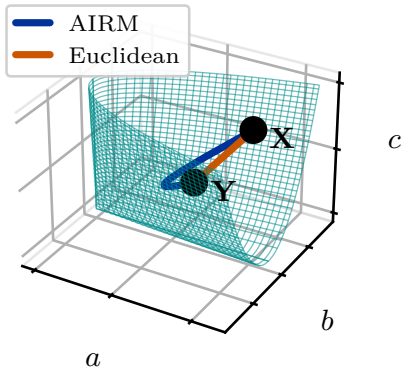


Figure D.7. L'espace $\mathcal{P}_2$ des matrices SDP de taille 2×2.

D.7.1.1 Une version riemannienne du t-SNE

Nous décrivons la version riemannienne du t-SNE. Étant donné N points $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \in \mathcal{P}_d$, on cherche $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_N \in \mathcal{P}_2$ qui préservent la structure locale : le voisinage de chaque point en basse dimension doit coïncider avec son voisinage d'origine.

En haute dimension, la similarité de $\boldsymbol{X}_j$ à $\boldsymbol{X}_i$ est donnée par la gaussienne isotrope riemannienne de Said, Bombrun et al. (2017) :

$$p_{j|i} = \frac{\exp\left(-\delta(\boldsymbol{X}_i, \boldsymbol{X}_j)^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\delta(\boldsymbol{X}_i, \boldsymbol{X}_k)^2 / 2\sigma_i^2\right)}, \tag{D.3}$$

où σ_i est ajusté par recherche dichotomique pour atteindre la perplexité $\texttt{perp} = 2^{H(P_i)}$ choisie par l'utilisateur. On symétrise : $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$ pour $i \neq j$.

En basse dimension, on utilise une loi de Student à un degré de liberté, dont les queues lourdes éloignent les points dissemblables :

$$q_{ij} = \frac{\left(1 + \delta(\boldsymbol{Y}_i, \boldsymbol{Y}_j)^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \delta(\boldsymbol{Y}_k, \boldsymbol{Y}_l)^2\right)^{-1}}. \tag{D.4}$$

L'écart entre (p_{ij}) et (q_{ij}) est mesuré par la divergence de Kullback-Leibler :

$$C = \sum_{i=1}^{N} \mathrm{KL}(P_i \,\|\, Q_i) = \sum_{i=1}^{N} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{D.5}$$

minimisée par descente de gradient riemannienne (Boumal 2023) sur la variété produit $(\mathcal{P}_2)^N$. L'algorithme est illustré à la figure D.8.

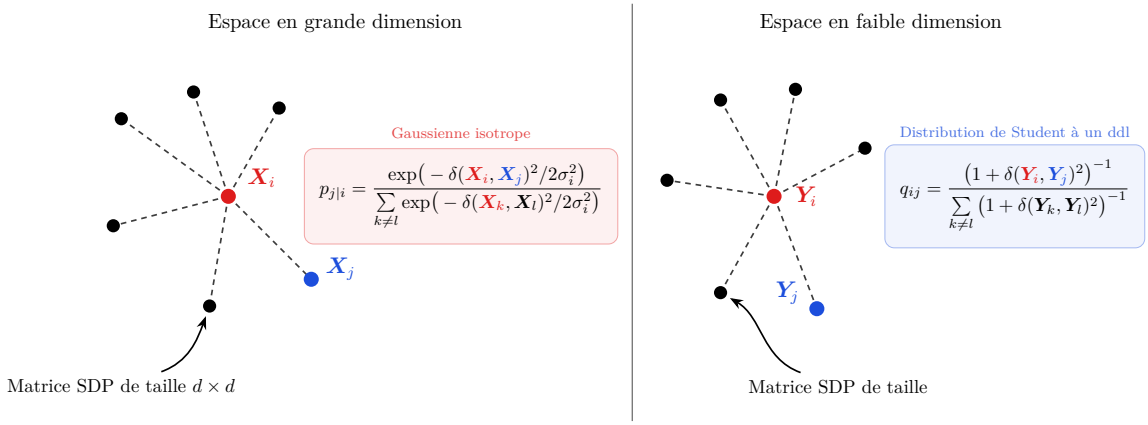


Figure D.8. Illustration du t-SNE riemannien.

D.7.1.2 Accélérer l'implémentation du t-SNE riemannien

Un soin particulier a été apporté à l'accélération de l'implémentation du t-SNE riemannien. En effet, si nous voulons que cet outil soit utilisé par d'autres chercheurs, il faut qu'il soit le plus rapide et le plus efficace possible. Une des clés de cette accélération est de remarquer que l'optimisation du t-SNE riemannien se fait sur l'espace produit $(\mathcal{P}_2)^N$: tous les calculs portent donc sur des matrices SDP de taille 2×2. Lorsque l'on utilise la métrique AIRM (Pennec 2020), ce qui est le cas ici, les opérations riemanniennes reposent sur l'extraction de valeurs propres et de vecteurs propres. Or, dans le cas des matrices de taille 2×2, les valeurs propres sont données explicitement par la proposition suivante :

Proposition D.4. *Soit* $\boldsymbol{X} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ *une matrice de taille* 2×2. *Alors, ses valeurs propres (éventuellement complexes) sont :*

$$\lambda_+ = \frac{a + d + \sqrt{(a-d)^2 + 4bc}}{2} \qquad \lambda_- = \frac{a + d - \sqrt{(a-d)^2 + 4bc}}{2}.$$

Cela permet d'accélérer grandement le calcul des opérations riemanniennes et donc le processus d'optimisation du t-SNE riemannien.

D'autres techniques, inspirées des nombreuses accélérations du t-SNE euclidien (Maaten et Hinton 2008 ; Belkina et al. 2019 ; Kobak et Berens 2019), sont également adaptées au t-SNE riemannien et permettent, ensemble, une optimisation rapide du t-SNE riemannien.

D.7.1.3 Expériences numériques

Nous validons le t-SNE riemannien sur six jeux de données réels (trois jeux d'ICO BNCI2014001/2/4, Air Quality, Textile et FPHA), en le comparant au *t-SNE riemannien-euclidien* (distances d'entrée AIRM, sortie dans $\mathbb{R}^3$) et au *t-SNE euclidien* (distance de Frobenius, sortie dans $\mathbb{R}^3$), tous deux issus de Scikit-Learn, ainsi qu'à UMAP, à l'ACP riemannienne et à la PGA. Empiriquement, la perplexité $\texttt{perp} = \frac{3}{4}N$ donne les meilleurs résultats pour notre méthode. La qualité des plongements est évaluée visuellement, puis par la *trustworthiness* (Venna et Kaski 2001). Ce score, compris entre 0 et 1, pénalise les faux voisins, et la variation de la taille du voisinage permet de sonder les structures locale et globale.

Résultats qualitatifs. Sur BNCI2014001, les méthodes tenant compte de la géométrie séparent bien les classes du sujet 8 (bon score de classification), contrairement au t-SNE euclidien, tandis que celles du sujet 2 (faible score) restent mélangées. Dans les deux cas,

les sessions *Train* et *Test* sont bien séparées, ce qui reflète un décalage de distribution entre sessions. De plus, le t-SNE riemannien est, avec l'ACP riemannienne, le seul à produire de véritables matrices SDP.

Résultats quantitatifs. Calculer les distances d'entrée avec l'AIRM améliore nettement le t-SNE. Le t-SNE riemannien-euclidien préserve bien la structure très locale ($< 10\,\%$ des voisins), mais le t-SNE riemannien est meilleur dès que le voisinage s'élargit, car la courbure déforme la variété à l'échelle globale. UMAP ne préserve que la structure locale, la PGA préserve la structure globale mais moins bien la structure locale (et fournit des vecteurs tangents, non des matrices SDP), et l'ACP riemannienne est nettement moins performante. Le t-SNE riemannien offre donc le meilleur compromis entre structures locale et globale.

D.7.2 Contribution de chaque électrode

Après l'outil de visualisation (t-SNE riemannien), nous cherchons à interpréter la sortie d'un classifieur riemannien : l'importance de chaque électrode est-elle cohérente avec la neurophysiologie ? Nous utilisons le jeu BNCI2014001 (Tangermann, K.-R. Müller et al. 2012) (9 sujets, 22 électrodes), restreint aux classes `left_hand` et `right_hand`. L'imagerie motrice des mains est classiquement décodée via la désynchronisation liée à l'événement (ERD) dans les bandes α et β au niveau des électrodes C3 et C4 (G. Pfurtscheller et Lopes da Silva 1999). Les sujets sont séparés en « bons » ($\geqslant 75\,\%$ de précision) et « mauvais » ($< 75\,\%$), et les résultats moyens sont obtenus avec le classifieur MDM (Barachant, Bonnet et al. 2011). Les méthodes présentées sont globales et agnostiques au modèle (post-hoc) (Molnar 2025).

D.7.2.1 Retrait d'une électrode

Pour chaque canal $j \in \{1, \dots, d\}$, on entraîne le classifieur sur les matrices de covariance *partielles* $\boldsymbol{X}_i^{(-j)}$ (ligne et colonne j supprimées) et on compare son score à celui du classifieur *complet* :

$$\frac{s^{(-j)} - s^{\text{full}}}{s^{\text{full}}}. \tag{D.6}$$

Une valeur négative indique une électrode utile, une valeur positive une électrode bruitée ou artefactuelle. Cette méthode a pour limite que le signal d'une source est capté par plusieurs électrodes voisines (faible résolution spatiale de l'EEG).

Résultats. Pour les bons sujets, presque toutes les électrodes sont utiles ; pour les mauvais, la plupart semblent apporter du bruit. Pour tous les sujets, C3 et C4 sont très utiles, en accord avec la neurophysiologie. Pour les bons sujets, certaines électrodes occipitales (P2, Pz, POz) paraissent aussi importantes, alors qu'elles couvrent le cortex visuel. Cela pourrait refléter un effet « Clever Hans » (Lapuschkin et al. 2019) (le classifieur exploiterait l'indice visuel), une imagerie visuelle ou une attention visuelle latéralisée (Christa Neuper et al. 2005).

D.7.2.2 Importance par permutation (PFI)

Le retrait compare des classifieurs entraînés sur des données différentes ($\mathcal{P}_d$ contre $\mathcal{P}_{d-1}$). La *Permutation Feature Importance* (PFI, Breiman 2001 ; A. Fisher et al. 2019) interprète au contraire un classifieur fixé. On l'entraîne sur $80\,\%$ des données, puis on permute aléatoirement le canal j entre les essais de test ($20\,\%$), on réestime les covariances et on mesure la baisse relative de score (moyenne sur 10 permutations et 10 découpages).

Résultats. Pour les bons sujets, C3 et C4 dominent clairement (chute de précision $> 20\,\%$), avec une légère asymétrie en faveur de l'hémisphère droit, mais le cortex visuel n'apparaît pas, donc cette méthode ne détecterait pas un éventuel effet « Clever Hans ». Pour les mauvais sujets, l'importance est uniformément faible : aucun canal ne contribue fortement à la décision.

Extension entièrement riemannienne. Permuter le canal j décorrèle celui-ci des autres : la ligne et la colonne j de la covariance deviennent quasi nulles, hormis l'élément diagonal σ_j^2. On peut donc simuler la permutation directement sur les covariances, sans accès à l'EEG brut, en annulant la ligne et la colonne j sauf l'élément (j, j). La matrice obtenue reste SDP : à permutation près, elle est diagonale par blocs, avec le bloc $\sigma_j^2 > 0$ et une sous-matrice principale de la covariance originale (SDP) ; or une matrice diagonale par blocs dont les blocs sont SDP est SDP. Les résultats sont très proches de la PFI classique pour un coût moindre, puisque les covariances ne sont estimées qu'une fois.

D.8 Discussion et perspectives

Dans le dernier chapitre de cette thèse, nous commençons par résumer les travaux présentés. Nous avons répondu à la question principale posée dans l'introduction grâce à la définition et à l'étude de la *gaussienne enveloppée*, principal outil théorique de cette thèse. Cette distribution est d'abord étudiée théoriquement, notamment en mettant l'accent sur la question de l'identifiabilité de la distribution et de l'estimation de ses paramètres. Puis, elle est utilisée pour construire un cadre probabiliste regroupant différents outils d'apprentissage sur les matrices SDP. Ce cadre rassemble des classifieurs, des outils de détection d'observations aberrantes, des noyaux et des méthodes de flots génératifs. Nous avons également montré que nous pouvions construire de nouveaux outils en nous basant sur la gaussienne enveloppée. Cela répond à la deuxième question posée en introduction. Enfin, nous répondons à la dernière question en appliquant une de ces méthodes à un problème réel en ICO, la détection de réveil lors d'une anesthésie. Nous avons montré que, grâce à l'anisotropie de la gaussienne enveloppée, nous étions capables de mieux distinguer les EEG enregistrés alors que le patient était réveillé de ceux enregistrés sous anesthésie. Le dernier chapitre de cette thèse est consacré à une question nouvelle portant sur l'interprétabilité des outils riemanniens utilisés en ICO. Nous proposons un outil de visualisation ainsi que des outils permettant de mieux comprendre l'influence de chaque électrode sur la prise de décision. Ce chapitre est une ouverture sur une question importante en apprentissage automatique, et la capacité d'expliquer une décision est cruciale pour démocratiser les ICO.

D.8.1 Limitations

Ce travail possède évidemment des limites. La principale est la complexité du modèle de la gaussienne enveloppée, dont le nombre total de paramètres est

$$\underbrace{\frac{d(d+1)}{2}}_{\boldsymbol{P}} + \underbrace{\frac{d(d+1)}{2}}_{\mu} + \underbrace{\frac{d^2(d+1)^2 + 2d(d+1)}{8}}_{\Sigma} = \mathcal{O}(d^4).$$

Cette complexité est inhérente au modèle que l'on construit et affecte principalement l'estimation des paramètres ainsi que l'utilisation de ce modèle dans des applications réelles. Une autre limite est la complexité numérique de la gaussienne enveloppée, qui repose sur le calcul d'exponentielles et de logarithmes riemanniens. Lorsque la géométrie AIRM est utilisée, ces opérations nécessitent des décompositions en valeurs propres, ce qui peut être lourd numériquement. Enfin, nous discutons également les limites des outils d'interprétabilité présentés.

D.8.2 Perspectives

Ce travail ouvre de nombreuses perspectives, à plus ou moins long terme. Certaines d'entre elles sont très liées à l'étude de la gaussienne enveloppée, telles que la construction de meilleurs estimateurs pour les paramètres, la résolution de la question de l'identifiabilité sur des variétés d'Hadamard générales, la construction de tests statistiques sur l'appartenance d'un point à une gaussienne enveloppée donnée ou l'étude d'une mixture de gaussiennes enveloppées. D'autres se focalisent sur les autres outils développés dans ces travaux, telles qu'une implémentation en ligne du t-SNE riemannien, qui permettrait de faire de la rétroaction pour des ICO, ou l'étude d'attaques adversariales dans un but d'interprétabilité.

Enfin, des perspectives à plus long terme sont également évoquées, autour des statistiques géométriques et des données structurées, par exemple l'étude d'outils de comparaison de distributions définies sur des espaces très structurés tels que des variétés. Pour les perspectives liées aux ICO, de la même manière que cela est fait pour les grands modèles de langage ou les modèles de fondation, avoir accès à d'énormes bases de données d'EEG pourrait permettre de mieux appréhender les variabilités affectant les ICO.

RÉSUMÉ

L'objectif de cette thèse est de construire une distribution de probabilité utile qui prenne en compte la géométrie riemannienne de l'espace $\mathcal{P}_d$ des matrices symétriques définies positives. Pour y parvenir, nous construisons une *gaussienne enveloppée* en transportant une gaussienne multivariée euclidienne depuis un plan tangent de $\mathcal{P}_d$ sur la variété. Cette distribution est étudiée théoriquement et l'estimation de ses paramètres est discutée. Ensuite, nous utilisons plusieurs variantes de la gaussienne enveloppée pour définir un cadre probabiliste englobant des outils d'apprentissage sur $\mathcal{P}_d$, comme des classifieurs, des noyaux, des algorithmes de détection d'observations aberrantes et des méthodes de flots génératifs riemanniens. En utilisant la flexibilité de la gaussienne enveloppée, nous étendons également certaines de ces méthodes afin d'obtenir des outils plus généraux et complexes. Nous montrons l'intérêt de ces outils sur une application spécifique aux Interfaces Cerveau-Ordinateur (ICO), dans laquelle nous cherchons à détecter le réveil au cours d'une anesthésie générale. Enfin, nous construisons des méthodes d'interprétabilité pour les outils riemanniens utilisés en ICO afin de mieux appréhender les décisions prises par les classifieurs riemanniens.

MOTS CLÉS

Statistiques géométriques, Matrices Symétriques Définies Positives, Gaussienne Riemannienne, Géométrie Riemannienne, Apprentissage Automatique, Interfaces Cerveau-Ordinateur.

ABSTRACT

The goal of this thesis is to build a useful geometry-aware probability distribution on the Riemannian manifold $\mathcal{P}_d$ of Symmetric Positive Definite matrices. In order to achieve this, we build a *wrapped Gaussian*, by mapping the multivariate Gaussian distribution from a tangent space of $\mathcal{P}_d$ to the manifold. This distribution is studied theoretically and the estimation of its parameters is discussed. Then, we use different variants of the Riemannian Gaussian to define a probabilistic framework encompassing learning tools on $\mathcal{P}_d$, such as classifiers, kernels, outlier detection algorithms and flow matching methods. Using the flexibility of the wrapped Gaussian, we are also able to extend some of these methods to more general and complex tools. We then show the interest of such tools on a specific application in Brain-Computer Interfaces (BCI), where we seek to detect wakefulness during general anesthesia. Finally, we build some interpretability methods for Riemannian tools used in BCI to better apprehend the decision taken by Riemannian classifiers.

KEYWORDS

Geometric statistics, Symmetric Positive Definite matrices, Riemannian Gaussian, Riemannian Geometry, Machine Learning, Brain Computer Interfaces.

