Geomstats: A Python Package for Riemannian Geometry in Machine Learning
Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, et al.

To cite this version:

HAL Id: hal-02536154
https://inria.hal.science/hal-02536154v2
Submitted on 21 Dec 2020

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

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
Geomstats: A Python Package for Riemannian Geometry in Machine Learning

Nina Miolane
NMIOLANE@STANFORD.EDU
Nico Guigui
NICOLAS.GUIGUI@INRIA.FR
Alice Le Brigant
ALICE.LE-BRIGANT@UNIV-PARIS1.FR
Johan Mathe
JOHAN@FROGLABS.AI
Benjamin Hou
BENJAMIN.HOU11@IMPERIAL.AC.UK
Yann Thanwerdas
YANN.TANWERDAS@INRIA.FR
Stefan Heyder
STEFAN.HEYDER@TU-ILMENAU.DE
Olivier Peltre
OPELITRE@GMAIL.COM
Niklas Koep
NIKLAS.KOEP@GMAIL.COM
Hadi Zaatiti
HADI.ZAATITI@IRT-SYSTEMX.FR
Hatem Hajri
HATEM.HAJRI@IRT-SYSTEMX.FR
Yann Cabanes
YANN.CABANES@GMAIL.COM
Thomas Gerald
THOMAS.GERALD@LIP6.FR
Paul Chauchat
PCHAUCHAT@GMAIL.COM
Christian Shewmake
CSHEWMAKE2@GMAIL.COM
Daniel Brooks
JIMRIVERS75@GMAIL.COM
Bernhard Kainz
B.KAINZ@IMPERIAL.AC.UK
Claire Donnat
CDONNAT@STANFORD.EDU
Susan Holmes
SUSAN@STAT.STANFORD.EDU
Xavier Pennec
XAVIER.PENNEC@INRIA.FR

Editor: Cheng Soon Ong

Abstract

We introduce Geomstats, an open-source Python package for computations and statistics on nonlinear manifolds such as hyperbolic spaces, spaces of symmetric positive definite matrices, Lie groups of transformations, and many more. We provide object-oriented and extensively unit-tested implementations. Manifolds come equipped with families of Riemannian metrics with associated exponential and logarithmic maps, geodesics, and parallel transport. Statistics and learning algorithms provide methods for estimation, clustering, and dimension reduction on manifolds. All associated operations are vectorized for batch computation and provide support for different execution backends—namely NumPy, PyTorch, and TensorFlow. This paper presents the package, compares it with related libraries, and provides relevant code examples. We show that Geomstats provides reliable building blocks to both foster research in differential geometry and statistics and democratize the use of Riemannian geometry in machine learning applications. The source code is freely available under the MIT license at geomstats.ai.

Keywords: differential geometry, Riemannian geometry, statistics, machine learning, manifold
1. Introduction

Data on manifolds naturally arise in different fields. **Hyperspheres** model directional data in molecular and protein biology (Kent and Hamelryck, 2005) and are used through Principal Nested Spheres to model some aspects of 3D shapes (Jung et al., 2012; Hong et al., 2016). Computations on **hyperbolic spaces** arise for impedance density estimation (Huckemann et al., 2010), geometric network comparison (Asta and Shalizi, 2014), and the analysis of reflection coefficients extracted from a radar signal (Chevallier et al., 2015). **Symmetric positive definite (SPD) matrices** characterize data from diffusion tensor imaging (DTI) (Pennec et al., 2006; Yuan et al., 2012) and functional magnetic resonance imaging (fMRI) (Sporns et al., 2005). Covariance matrices, which are also SPD matrices, appear in applications such as automatic speech recognition (ASR) systems (Shinohara et al., 2010), image and video descriptors (Harandi et al., 2014), or air traffic complexity representation (Brigant and Puechmorel, 2019). The **Lie groups of transformations** SO(3) and SE(3) appear naturally when dealing with articulated objects like the human spine (Arsigny, 2006; Boisvert et al., 2006) or the pose of a camera (Kendall and Cipolla, 2017; Hou et al., 2018). **Stiefel manifolds** are used to process video action data or to analyze two vector-cardiograms (Chakraborty and Vemuri, 2019). **Grassmannians** appear in computer vision to perform video-based face recognition and shape recognition (Turaga et al., 2008).

Statistics on **landmark spaces** are used in anthropology and many applied fields to describe biological shapes (Richtsmeier et al., 1992). More generally, Kendall shape spaces gave rise to a very important literature in shape statistics (Dryden and Mardia, 1998). A variety of applications use tools from infinite-dimensional Riemannian geometry to study the shapes of **discretized curves** such as those sampled from closed two or three-dimensional curves defining the contours of organs in computational anatomy (Younes, 2012). In addition, open curves that describe the temporal evolution of physical phenomena are embedded in various manifolds, for example in a Lie group (Celledoni et al., 2015) or in the hyperbolic plane (Le Brigant, 2017).

Yet, the adoption of methods from differential geometry has been inhibited by the lack of a reference implementation. Code sequences are often custom-tailored for specific problems and are not easily reused. Some Python packages do exist, but they often focus on optimization: **Pymanopt** (Townsend et al., 2016), **Geoopt** (Bécigneul and Ganea, 2018; Kochurov et al., 2019), and **McTorch** (Meghwanshi et al., 2018). Others are dedicated to a single manifold: **PyRiemann** on SPD matrices (Barachant et al., 2015), **PyQuaternion** on 3D rotations (Wynn, 2014), and **PyGeometry** on spheres, toruses, 3D rotations and translations (Censi, 2012). Lastly, others lack unit-tests and continuous integration: **TheanoGeometry** (Kühnel and Sommer, 2017). There is a need for an open-source implementation of differential geometry and associated learning algorithms for manifold-valued data.

We present **Geomstats**, an open-source Python package for computations and statistics on nonlinear manifolds. **Geomstats** has three main objectives: (i) foster research in differential geometry and geometric statistics by providing low-level code to gain intuition or test a theorem and a platform to share algorithms; (ii) democratize the use of geometric statistics by implementing user-friendly geometric learning algorithms using Scikit-Learn API; and (iii) provide educational support to learn “hands-on” differential geometry and geometric statistics, through its examples, notebooks and visualizations.
2. Implementation Overview

The package geomstats is organized into two main modules: geometry and learning. The module geometry implements concepts in Riemannian geometry with an object-oriented approach. Manifolds mentioned in the introduction are available as classes that inherit from the base class Manifold. The base class RiemannianMetric provides methods such as the geodesic distance between two points, the exponential and logarithm maps at a base point, etc. As examples, HyperbolicMetric, StiefelCanonicalMetric, the Lie groups’ InvariantMetrics, or the curves’ L2Metric and SRVMetric (Srivastava et al., 2011) inherit from RiemannianMetric. Going beyond Riemannian geometry, the class Connection implements affine connections. The implementation uses automatic differentiation with autograd to allow computations on manifolds when closed-form formulae do not exist. The API of the module geometry uses terms from differential geometry to foster contributions of researchers from this field. The module learning implements statistics and learning algorithms for data on manifolds. The code is object-oriented and classes inherit from Scikit-Learn base classes and mixins: BaseEstimator, ClassifierMixin, RegressorMixin, etc. This module provides classes such as FrechetMean, KMeans, TangentPCA, for implementations of Fréchet mean estimators (Fréchet, 1948), K-means, and principal component analysis (PCA) designed for manifold data. The API of the module learning follows Scikit-Learn’s API, being therefore user-friendly to machine learning researchers and engineers.

The code follows international standards for readability and ease of collaboration, is vectorized for batch computations, undergoes unit-testing with continuous integration using Travis, relies on either Numpy, TensorFlow or PyTorch backends. The package comes with a visualization module to provide intuition on differential geometry (see Figure 1), and with a datasets module that provides toy data sets on manifolds. The repositories examples and notebooks provide convenient starting points to get familiar with geomstats.

The GitHub repository at github.com/geomstats/geomstats offers a convenient way to ask for help or request features by raising issues. The website geomstats.github.io provides documentation for users and important guidelines for those wishing to contribute to the project.

3. Comparison and Interaction with Existing Packages

This section compares the package geomstats with related Python implementations on differential geometry and learning. Table 1 compares the geometric operations and Table 2 compares the engineering infrastructures.

The library TheanoGeometry (Kühnel and Sommer, 2017) is the most closely related to GEOMSTATS and provides nonlinear statistics and stochastic equations on Riemannian manifolds. The differential geometric tensors are computed with automatic differentiation. However, this library does not provide statistical learning algorithms and lacks engineering maintenance. Several other packages focus on optimization on Riemannian manifolds. Pymanopt (Townsend et al., 2016) computes gradients and Hessian-vector products on Riemannian manifolds with automatic differentiation and provides the following solvers: steepest descent, conjugate gradient, the Nelder-Mead algorithm, particle swarm optimization, and the Riemannian trust regions. Geoopt (Kochurov et al., 2019) focuses on stochastic adap-
Figure 1: Left: Minimization of a scalar field on the sphere $S^2$ using Riemannian gradient descent. Middle: Regular geodesic grid on the hyperbolic space $\mathbb{H}^2$ in Poincaré disk representation. Right: Geodesic on the Lie group $SE(3)$ for the canonical left-invariant metric. These examples and more are available at geomstats.ai.

ative optimization on Riemannian manifolds, for machine learning problems. The library provides stochastic solvers, stochastic gradient descent and Adam, as well as the following samplers: Stochastic Gradient Langevin Dynamics, Hamiltonian Monte-Carlo, Stochastic Gradient Hamiltonian Monte-Carlo. Lastly, McTorch (Meghwanshi et al., 2018) provides optimization on Riemannian manifold for deep learning by adding a “Manifold” parameter to PyTorch’s network layers and optimizers. The library provides the following solvers: stochastic gradient descent, AdaGrad and conjugate gradients. As these libraries focus on optimization, they substitute potentially computationally expensive operations by practical proxies, for example, by replacing exponential maps by so-called retractions. However, they are less modular than GEOMSTATS in terms of the Riemannian geometry. For example, each manifold comes with a single Riemannian metric, in contrast to GEOMSTATS where families of Riemannian metrics are implemented. Furthermore, they do not provide statistical learning algorithms.

The optimization libraries are complementary to GEOMSTATS and interact easily with it. GEOMSTATS provides low-level implementations of Riemannian geometry that can be used to define optimization costs. In turn, an optimization library can provide an efficient solver to use within the implementation of GEOMSTATS’ learning algorithms. An example of such interactions, between Pymanopt and GEOMSTATS can be found in GEOMSTATS’ examples folder.

4. Usage: Examples of Learning on Riemannian manifolds

Three steps are needed to run learning algorithms on manifolds with GEOMSTATS: (i) instantiate the manifold of interest, (ii) instantiate the learning algorithm of interest, and (iii) run the algorithm. The following code snippet illustrates the use of $K$-means on the hypersphere.

```python
sphere = Hypersphere(dim=5)
```
Geomstats: A Python Package for Riemannian Geometry in Machine Learning

- **Manifolds**: Euclidean manifold, symmetric matrices, Stiefel, Grassmannian, oblique manifold, SPD(n), ellipse, fixed-rank PSD matrices
- **Geometry**: Exponential and logarithmic maps, retraction, vector transport, $\text{egrad2rgrad}$, $\text{ehess2rhess}$, inner product, distance, norm

<table>
<thead>
<tr>
<th>Library</th>
<th>Manifolds</th>
<th>Geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pymanopt</td>
<td>Euclidean manifold, sphere, complex circle, SO($n$), Stiefel, Grassmannian, oblique manifold, SPD($n$), ellipse, fixed-rank PSD matrices</td>
<td>Exponential and logarithmic maps, retraction, vector transport, $\text{egrad2rgrad}$, $\text{ehess2rhess}$, inner product, distance, norm</td>
</tr>
<tr>
<td>Geoopt</td>
<td>Euclidean manifold, sphere, Stiefel, Poincaré ball</td>
<td>Same as Pymanopt</td>
</tr>
<tr>
<td>McTorch</td>
<td>Stiefel, SPD($n$)</td>
<td>Same as Pymanopt</td>
</tr>
<tr>
<td>TheanoGeometry</td>
<td>Sphere, ellipsoid, SPD($n$), Landmarks, GL($n$), SO($n$), SE($n$)</td>
<td>Inner product, exponential and logarithmic maps, parallel transport, Christoffel symbols, Riemann, Ricci and scalar curvature, geodesics, Fréchet mean</td>
</tr>
<tr>
<td>Geomstats</td>
<td>Euclidean, Minkowski, hyperbolic space, Poincaré polydisk, hypersphere, SO($n$), SE($n$), GL($n$), Stiefel, Grassmannian, SPD($n$), symmetric matrices, skew-symmetric matrices, discretized curves on manifolds, landmarks on manifolds</td>
<td>Levi-Civita connection, Christoffel symbols, parallel transport, exponential and logarithmic maps, inner product, distance, norm, geodesics, group invariant metrics, Fréchet means and learning algorithms on manifolds</td>
</tr>
</tbody>
</table>

Table 1: Comparison of libraries in terms of geometric operations

<table>
<thead>
<tr>
<th>Backends</th>
<th>Continuous integration (CI) and coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pymanopt</td>
<td>CI, coverage 85%</td>
</tr>
<tr>
<td>Geoopt</td>
<td>CI, coverage 84%</td>
</tr>
<tr>
<td>McTorch</td>
<td>CI, coverage 85%</td>
</tr>
<tr>
<td>TheanoGeometry</td>
<td>CI, coverage 85%</td>
</tr>
<tr>
<td>Geomstats</td>
<td>CI, coverage 92% (NumPy), 76% (TensorFlow), 79% (PyTorch)</td>
</tr>
</tbody>
</table>

Table 2: Comparison of code infrastructure
data = sphere.random_uniform(n_samples=10)
clustering = OnlineKMeans(metric=sphere.metric, n_clusters=4)
clustering = clustering.fit(data)

The following code snippet shows the use of tangent PCA on the 3D rotations.

so3 = SpecialOrthogonal(n=3, point_type='vector')
metric = so3.bi_invariant_metric
data = so3.random_uniform(n_samples=10)
trca = TangentPCA(metric=metric, n_components=2)
trca = trca.fit(data)
tangentProjectedData = trca.transform(data)

All geometric computations are performed behind the scenes. The user only needs a
high-level understanding of Riemannian geometry. Each algorithm can be used with any of
the manifolds and metric implemented in the package. The folders examples and notebooks
provide many more code snippets that help users get started with Geomstats.

5. Conclusion

We presented the Python Package Geomstats, with the aim to provide the wider Machine
Learning community with off-the-shelf geometric learning algorithms on a wide variety of
manifolds, and flexibility in the choice of metrics, while being faithful to the mathematician’s
formulation of Riemannian geometry. This sometimes comes at cost of efficiency, and future
contributions will be devoted to addressing this caveat.

Acknowledgments

This work is partially supported by the National Science Foundation, grant NSF DMS RTG
1501767, the Inria-Stanford associated team GeomStats, the European Research Council
(ERC) under the EU Horizon 2020 research and innovation program (grant agreement G-
Statistics No. 786854) and by the French Government through the 3IA Côte d’Azur Invest-
ments in the Future project (National Research Agency ANR-19-P3IA-0002).

References

Vincent Arsigny. Processing Data in Lie Groups: An Algebraic Approach. Application to
Non-Linear Registration and Diffusion Tensor MRI. PhD thesis, École polytechnique, 11
2006.


Alexandre Barachant. PyRiemann: Python package for covariance matrices manipulation
and Biosignal classification with application in Brain Computer interface, 2015. URL
https://github.com/alexandrebarachant/pyRiemann.


