Journal of Machine Learning Research 22 (2021) 1-6 Submitted 4/20; Revised 11/20; Published 1/21

giotto-tda: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration

Guillaume Tauzin[†] GTAUZIN@PRONTONMAIL.COM
Umberto Lupo[§] UMBERTO.LUPO@GMAIL.COM
Lewis Tunstall[§] LEWIS.C.TUNSTALL@GMAIL.COM
Julian Burella Pérez[‡] JULIAN.BURELLAPEREZ@HEIG-VD.CH
Matteo Caorsi[§] M.CAORSI@L2F.CH
Anibal M. Medina-Mardones[†] ANIBAL.MEDINAMARDONES@EPFL.CH
Alberto Dassatti[‡] ALBERTO.DASSATTI@HEIG-VD.CH
Kathryn Hess[†] KATHRYN.HESS@EPFL.CH
[†] *Laboratory for Topology and Neuroscience, EPFL*
[§] *L2F SA*
[‡] *School of Management and Engineering Vaud, HES-SO*

Editor: Alexandre Gramfort

Abstract

We introduce *giotto-tda*, a `Python` library that integrates high-performance topological data analysis with machine learning via a *scikit-learn*–compatible API and state-of-the-art `C++` implementations. The library's ability to handle various types of data is rooted in a wide range of preprocessing techniques, and its strong focus on data exploration and interpretability is aided by an intuitive plotting API. Source code, binaries, examples, and documentation can be found at `https://github.com/giotto-ai/giotto-tda`.

Keywords: Topological Data Analysis, Persistent Homology, Mapper, Machine Learning, Data Exploration, Python

1. Introduction

Topological data analysis (TDA) uses tools from algebraic and combinatorial topology to extract features that capture the shape of data (Carlsson, 2009). In recent years, algorithms based on topology have proven very useful in the study of a wide range of problems. In particular, *persistent homology* has had significant impact on data intensive challenges including the classification of porous materials (Lee et al., 2018), the study of structures in the weight space of CNNs (Gabrielsson and Carlsson, 2018), and the discovery of links between structure and function in the brain (Reimann et al., 2017). The *mapper* algorithm has also received considerable attention after its use in the identification of a highly treatable subgroup of breast cancers (Nicolau et al., 2011).

Despite its power and versatility, TDA has remained outside the toolbox of most machine learning (ML) practitioners, largely because current implementations are developed for research purposes and not in high-level languages. The aim of *giotto-tda* is to fill this gap by making TDA accessible to the `Python` data science community, while supporting research. To this end, *giotto-tda* inherits the flexibility of *scikit-learn*, the most popular all-

purpose ML framework (Pedregosa et al., 2011), and extends it with TDA capabilities that include a wide range of persistent homology and Mapper-type algorithms. It enables TDA to be applied to univariate and multivariate time series, images, graphs, and their higher dimensional analogues, simplicial complexes. This makes *giotto-tda* the most comprehensive `Python` library for topological *machine learning* and data exploration to date.

2. Architecture

To use topological features in machine learning effectively, techniques such as hyperparameter search and feature selection need to be applied at a large scale. Facilitating these processes is one of the reasons why *giotto-tda* maintains and extends compatibility with the *scikit-learn* API. *giotto-tda* provides users with full flexibility in the design of TDA pipelines via modular `estimators`, and the highly visual nature of topological signatures is harnessed via a plotting API based on *plotly*. This exposes a set of external functions and class methods to plot and interact with intermediate results represented as standard *NumPy* arrays (Harris et al., 2020).

To apply TDA techniques to time series, one must first embed the input data into a higher-dimensional space. To support flexible embedding options while maintaining high levels of integration with *scikit-learn*, *giotto-tda* defines a `TransformerResamplerMixin` base class. It provides a `resample` method that modifies the target's sample number to align it with the transformed input data. For users to be able to combine *scikit-learn*–based `estimators` and *giotto-tda*'s `transformer-resamplers`, an extended version of *scikit-learn*'s `Pipeline` is provided.

3. Persistent Homology

Persistent homology is one of the main tools in TDA. It extracts and summarises, in so-called persistence diagrams, multi-scale relational information in a manner similar to hierarchical clustering, but also considering higher-order connectivity. To fully take advantage of it in ML and data exploration tasks, *giotto-tda* offers *scikit-learn*–compatible components that enable the user to a) transform a wide variety of data input types into forms suitable for computing persistent homology, b) compute persistence diagrams according to a large selection of algorithms, and c) extract a rich set of features from persistence diagrams. The result is a framework for constructing end-to-end `Pipeline` objects to generate carefully crafted topological features from each sample in an input raw data collection. At a more technical level, features are often extracted from persistence diagrams by first representing them as curves or images, or by defining kernels. Each method for doing so typically comes with a set of hyperparameters that must be tuned to the problem at hand. *giotto-tda* exposes a large selection of such algorithms and, by tightly integrating with the *scikit-learn* API for hyperparameter search, cross-validation and feature selection, allows for simple data-driven tuning of the many hyperparameters involved. A feature comparison between *giotto-tda* and other `Python` persistence homology libraries is shown in Table 1.

Our library matches the code and documentation standards set by *scikit-learn*, and relies on state-of-the-art external `C++` libraries (The GUDHI Project, 2020; Bauer, 2019; Kerber

1. *GUDHI* (The GUDHI Project, 2020), *scikit-tda* (Saul and Tralie, 2019), *Dionysus 2* (Morozov, 2018).

	ML framework integration	Preprocessing					Persistent homology			Diagrams features	Plotting
		Point clouds	Time series	Networks	Images	Persistent diagrams	Simplicial (undirected)	Simplicial (directed)	Cubical		
giotto-tda	✓✓	✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
GUDHI	✓	✓✓	✓	✗	✗	✓✓	✓✓	✗	✓✓	✓✓	✓
scikit-tda	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓
Dionysus 2	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✓

Table 1: Relative degree of support in the main `Python` persistent homology libraries.[1]

et al., 2017; Lütgehetmann et al., 2020) using new performance-oriented bindings based on *pybind11* (Jakob et al., 2017).[2] Whenever possible, we contributed with bug fixes and other improvements to *giotto-tda*'s `C++` and `Python` dependencies. For `flagser` (Lütgehetmann et al., 2020), no `Python` API was available prior to *giotto-tda*'s sibling project *pyflagser*.[3] An example of a *giotto-tda* persistent homology `Pipeline` for images of handwritten digits is shown in Figure 1.

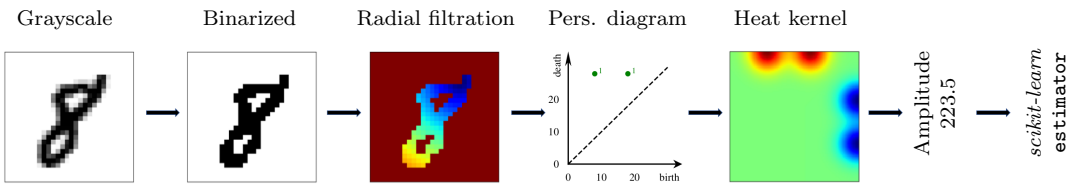

Figure 1: Example of a *giotto-tda* pipeline processing a MNIST image.[4]

4. Mapper

Mapper is a representation technique of high-dimensional data that, combining the application of filter functions and partial clustering, creates a simple and topologically meaningful description of the input as an unweighted graph (or, more generally, as a simplicial complex). It is primarily used as a data visualization tool to explore substructures of interest in data. In *giotto-tda*, this algorithm is realised as a sequence of steps in a *scikit-learn* `Pipeline`, where the clustering step can be parallelized. The resulting graph is visualized through an interactive plotting API. This design choice provides a great deal of interoperability and computational efficiency, allowing users to a) realize relevant steps of the Mapper algorithm through any *scikit-learn* `estimator`, b) integrate Mapper pipelines as part of a larger ML workflow, and c) make use of memory caching to avoid unnecessary re-computations. Memory caching is especially useful for interactive plotting, where *giotto-tda* allows users to tune mapper's hyperparameters and observe how the resulting graph changes in real time. An example is shown in Figure 2.

To the best of our knowledge, *KeplerMapper* (van Veen et al., 2019) is the only alternative open-source implementation of Mapper in `Python` that provides general-purpose functionality. Although *KeplerMapper* also provides the flexibility to use *scikit-learn* `estimators` to generate mapper graphs, it does not implement all steps of the algorithm in a single class

2. In the case of `ripser` (Bauer, 2019), bindings from `ripser.py` (Tralie et al., 2018) were adapted.
3. Source code available at `https://github.com/giotto-ai/pyflagser`.
4. Figure adapted from Garin and Tauzin (2019).

and is only partially compatible with *scikit-learn* `pipelines`. Moreover, it does not implement memory caching or provide real-time hyperparameter interactivity in the visualization.

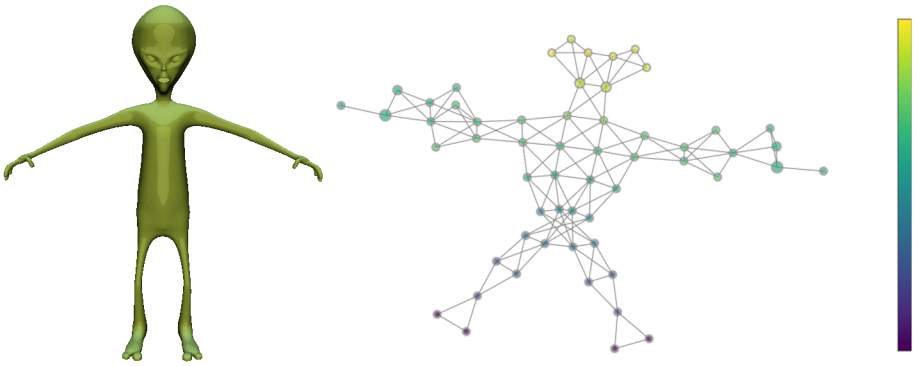

Figure 2: Mapper graph generated by *giotto-tda* based on the height of a 3D model.[5]

5. Project Management

Installation: Binary packages are available for all major operating systems on the PyPI package repository and can be installed using `python -m pip install -U giotto-tda`.
Code quality: The code is unit-tested throughout using *pytest* and *hypothesis* and test coverage is at 98%. The code follows `PEP8` standards and adheres to the `Python` coding guideline and *NumPy*-style documentation.
Community-based development: We base *giotto-tda*'s development on collaborative tools such as Git, GitHub, and Slack. Contributions are encouraged, and we actively make use of GitHub's issue tracker to provide support and discuss ideas. The library is distributed under the GNU AGPLv3 license.
Documentation and learning resources: A detailed API reference is provided to the user using *sphinx*.[6] To lower the entry barrier, we provide a theory glossary and a wide range of tutorials and examples that help new users explore how TDA-based ML pipelines can be applied to data sets of various sorts.
Project relevance: As of v0.3.1, the GitHub repository has attracted over 300 stars and between 500 and 1000 visits per week. The PyPI package is downloaded 350 times per month. The library appears in *scikit-learn*'s curated list of related projects.

6. Concluding Remarks

The very active research field of TDA provides algorithms that can be used at any step of a ML pipeline. *giotto-tda* aims to make these algorithms available in a form that is useful to both the research and data science communities, thus allowing them to use TDA as a part of large-scale ML tasks. We have written *giotto-tda* under the code and documentation standards of *scikit-learn* and, alongside further performance optimization of the existing `C++` code, future developments will include the first implementation of novel TDA algorithms such as persistence Steenrod diagrams (Medina-Mardones, 2018).

5. Example adapted from Murugan and Robertson (2019).
6. Currently hosted at `https://giotto-ai.github.io/gtda-docs/latest/modules/index.html`

Acknowledgments

We thank Roman Yurchak, Philippe Nguyen, Philipp Weiler, and Wojciech Reise for their ideas and contributions, and Innosuisse (grant number 32875.1 lP-ICT) for its support.

References

Ulrich Bauer. Ripser: efficient computation of Vietoris-Rips persistence barcodes, August 2019. Preprint.

Gunnar Carlsson. Topology and data. *Bull. Amer. Math. Soc. (N.S.)*, 46(2):255–308, 2009.

Rickard Brüel Gabrielsson and Gunnar Carlsson. Exposition and interpretation of the topology of neural networks, 2018.

Adélie Garin and Guillaume Tauzin. A topological "reading" lesson: Classification of MNIST using TDA. In *Proceedings of the 19th International Conference on Machine Learning and Applications (ICMLA 2020)*. IEEE, December 2019.

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, et al. Array programming with NumPy. *Nature*, 585:357–362, 2020.

Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. URL `https://github.com/pybind/pybind11`.

Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics*, 22:1–20, 09 2017.

Yongjin Lee, Senja D Barthel, Paweł Dłotko, et al. High-Throughput Screening Approach for Nanoporous Materials Genome Using Topological Data Analysis: Application to Zeolites. *Journal of chemical theory and computation*, 14(8):4427–4437, August 2018.

Daniel Lütgehetmann, Dejan Govc, Jason P. Smith, et al. Computing persistent homology of directed flag complexes. *Algorithms*, 13(1), 2020.

A. M. Medina-Mardones. Persistence Steenrod modules, 2018. URL `https://arxiv.org/abs/1812.05031`.

Dmitriy Morozov. Dionysus 2 – library for computing persistent homology, 2018. URL `https://github.com/mrzv/dionysus`.

Jeff Murugan and Duncan Robertson. An introduction to topological data analysis for physicists: From lgm to frbs, 2019.

Monica Nicolau, Arnold J. Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Michael W. Reimann, Max Nolte, Martina Scolamiero, et al. Cliques of neurons bound into cavities provide a missing link between structure and function. *Frontiers in Computational Neuroscience*, 11:48, 2017.

Nathaniel Saul and Chris Tralie. Scikit-tda: Topological data analysis for python, 2019. URL `https://doi.org/10.5281/zenodo.2533369`.

The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 3.1.1 edition, 2020. URL `https://gudhi.inria.fr/doc/3.1.1/`.

Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A lean persistent homology library for python. *Journal of Open Source Software*, 3(29):925, 2018. doi: 10.21105/joss.00925. URL `https://doi.org/10.21105/joss.00925`.

Hendrik van Veen, Nathaniel Saul, David Eargle, et al. Kepler Mapper: A flexible Python implementation of the mapper algorithm. *Journal of Open Source Software*, 4(42):1315, 2019. URL `https://github.com/scikit-tda/kepler-mapper`.