The Journal of Open Source Software

TDAvec: Computing Vector Summaries of Persistence Diagrams for Topological Data Analysis in R and Python

Umar Islambekov [1*] **and Aleksei Luchinsky** [1*¶]

1 Bowling Green State University, USA ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

The theory of *persistent homology* is one of the popular tools in *topological data analysis* (TDA) to analyze data with underlying shape structure (Carlsson, 2009; Chazal & Michel, 2021; Edelsbrunner & Harer, 2010). In this context, a single data observation could be a collection of points lying in a metric space, an image, a graph or a time series. The basic idea behind persistent homology is to build a nested sequence (or *filtration*) of *simplicial complexes* (indexed by a scale parameter) on top of data points and keep a record of the appearance and disappearance of various topological features at different scale values. Here, these topological features are "holes" of different dimensions – connected components, loops, voids, and their higher-dimensional versions whose emergence and subsequent disappearance are tracked using a concept of homology from algebraic topology. From a geometric point of view, simplicial complexes consist of vertices, edges, triangles, tetrahedra etc., glued together and serve as a means for recovering (at least partially) the underlying shape information which is lost during sampling (Nanda & Sazdanovic, 2013).

A topological descriptor outputted by the persistent homology encoding the shape of data is called a *persistence diagram* (PD). Mathematically, a k-dimensional PD is a multi-set of points $D = \{(b_i, d_i)\}_{i=1}^N$, where each point (b_i, d_i) corresponds to a topological feature of homological dimension k (0 if a connected component, 1 if a loop, 2 if a void, etc) with the x-coordinate representing the scale at which this feature is born (or created), and the y-coordinate representing the scale at which it dies (or disappears). In practice, one is usually interested in applying a machine learning method to PDs to make further inferences from data. However, the fact that PDs do not form a Hilbert space, which is a feature (or an input) space for a wide class of machine learning methods, limits their direct of use in applications. To overcome this challenge, kernel methods and vectorization techniques are commonly used (Chung & Lawson, 2022). The kernel approach involves defining a notion of similarity between pairs of PDs, whereas the vectorization methods aim to transform PDs into finite-dimensional feature vectors that can be used as input for many standard machine learning models. In recent years, the kernel and vectorization approaches have proven successful and gained prominence in the applied TDA literature (see Hensel et al. (2021) for a survey of applications of TDA in machine learning).

R tools for TDA

The computational tools for TDA in the R environment are provided through various packages[1] such as `TDA` (Fasy et al., 2021), `TDAstats` (R. R. Wadhwa et al., 2018), `kernelTDA` (Padellini & Palini, 2020), `TDAmapper` (Pearson et al., 2015), `TDAkit` (You & Yu, 2021), `tdaunif` (Brunson et al., 2024), `TDApplied` (Brown & Farivar, 2024) and `ripserr` (R. Wadhwa et al., 2020).

[1]In this overview, we only focus on R packages for TDA that are available on the CRAN repository.

The TDA package is the largest R package for TDA. The TDA package offers tools to compute PDs for commonly used types of filtrations such as *Vietoris-Rips*, *Alpha* and *Alpha shape*. It also allows to construct more general sublevel set filtrations and compute the corresponding PDs. Moreover, the TDA package provides implementations to plot PDs and compute *bottleneck and Wasserstein* distances between them. TDAstats offers a variety of tools for conducting statistical inference (such as hypothesis testing) on PDs. Compared to the TDA package, it computes PDs much faster for Vietoris-Rips filtrations based on the Ripser C++ library (Bauer, 2021) and offers more aesthetic visualization of the diagrams using the ggplot2 package (Wickham, 2016). The kernelTDA package contains implementations of popular kernel-based methods for TDA such as *geodesic Gaussian kernel*, *geodesic Laplacian kernel*, *persistence Fisher kernel* and *persistence sliced Wasserstein kernel*. For computing the Wasserstein distance between a pair of PDs, unlike the TDA package, it uses an iterative procedure to reasonably approximate the exact distance which that leads to a considerable reduction in run-time cost. The ripserr package allows a fast computation of PDs for filtrations on Vietoris-Rips and cubical complexes using the Ripser C++ library. The TDApplied and TDAkit packages provides various tools to integrate topological features (PDs or their vector summaries) into machine and statistical learning settings. The tdaunif is a useful package if one needs to sample points from various manifolds such as a klein-bottle, an ellipse or a torus. TDAmapper offers tools to visualize high-dimensional data by constructing the so-called Mapper graphs that preserve its topological structure.

Python tools for TDA

Several Python libraries are available for TDA, such as Giotto-tda (Tauzin et al., 2021), Ripser (Christopher et al., 2018), Gudhi (Rouvreau, 2020), Scikit-tda, Dionysus 2 (Morozov, 2018), Persim (Saul, 2019) and KeplerMapper (Van Veen et al., 2019). Giotto-tda is a powerful library that integrates with the popular machine learning library scikit-learn, offering tools for persistent homology and visualizations of persistence diagrams. Ripser focuses on fast computation of Vietoris-Rips complexes, especially for large datasets. Gudhi provides a wide range of topological tools for simplicial complexes, persistent homology, and topological signatures. Scikit-tda is another package that integrates with scikit-learn, simplifying the application of TDA to typical machine learning tasks. Dionysus 2 offers fast computation of persistent homology and cohomology, with an emphasis on flexibility and efficiency. Persim focuses on tools for working with PDs. It contains implementations of commonly used vectorization and kernel methods for PDs. KeplerMapper implements the TDA Mapper algorithm to visualize high-dimensional data. For a more comprehensive list of Python libraries for TDA and their functionality, we refer the readers to (*Awesome-TDA*, 2024).

Statement of need

The problem of transforming PDs into finite dimensional vectors for machine learning purposes has attracted considerable attention in the TDA research community over the past decade. While early vectors summaries of PDs such as *persistence landscape* (Bubenik, 2015), *persistence silhouette* (Chazal et al., 2014), *Betti curve*[2] (Chazal & Michel, 2021) and *persistence image* (Adams et al., 2017) have been implemented in both Python and R packages, there is no single package that systematically gathers them - the early ones as well as those introduced in recent years - in one place under unified syntax. Moreover, in R, all the code behind the existing vector implementations is written using standard R functions which may prove slow and inefficient for large-scale computations. The TDAvec R package and its Python version aim to fill in some of these gaps. Its contributions can be summarized in the following three areas:

1. It expands the list of implemented vector summaries for PDs by providing vectorizations of 13 commonly used methods in TDA. These methods are grouped into three broad categories:

[2] also called *Betti function*

- Functional vector summaries - based on summary functions:
  - Betti curve (Chazal & Michel, 2021)
  - Euler characteristic curve (Richardson & Werman, 2014)
  - Normalized life curve (Chung & Lawson, 2022)
  - Persistence block (Chan et al., 2022)
  - Persistence surface (Adams et al., 2017)
  - Persistence landscape function (Bubenik, 2015)
  - Persistence silhouette function (Chazal et al., 2014)
  - Persistent entropy summary function (Atienza et al., 2020)
  - Template function (Perea et al., 2023)
- Algebraic vector summaries - based on polynomial maps:
  - Algebraic functions (Adcock et al., 2013)
  - Complex polynomial coefficients (Di Fabio & Ferri, 2015; Ferri & Landi, 1999)
  - Tropical coordinate function (Kališnik, 2019)
- Statistical vector summaries - based on descriptive statistics:
  - Basic descriptive statistics (Ali et al., 2023)

2. A univariate summary function f of a PD is commonly vectorized by evaluating it at a sequence of points on a one-dimensional grid, then organizing the resulting values into a vector:

$$(f(t_1), f(t_2), \dots, f(t_n)) \in \mathbb{R}^n, \tag{1}$$

   where $t_1, t_2, \dots, t_n$ form an increasing sequence of scale values. For instance, the `landscape()` and `silhouette()` functions in the `TDA` package produce such vector summaries for persistence landscapes and silhouettes, respectively. In addition to this standard approach, the `TDAvec` package introduces an alternative vectorization scheme that captures the average behavior of f between consecutive scale values t_i and t_{i+1} through integration:

$$\Big(\frac{1}{\Delta t_1}\int_{t_1}^{t_2} f(t), dt, \frac{1}{\Delta t_2}\int_{t_2}^{t_3} f(t), dt, \dots, \frac{1}{\Delta t_{n-1}}\int_{t_{n-1}}^{t_n} f(t), dt\Big) \in \mathbb{R}^{n-1}, \tag{2}$$

   where $\Delta t_i = t_{i+1} - t_i$. Unlike the method in (1), this approach retains information about the behavior of f between neighboring scale points. It is applicable to any univariate summary function that is integrable in closed form, such as the persistence silhouette, persistent entropy summary function, Euler characteristic curve, normalized life curve, and Betti function. Users have the flexibility to choose between the two vectorization methods based on their application needs.

3. To achieve higher computational efficiency, all code behind the vector summaries of `TDAvec` is written in C++ using the `Rcpp` and `RcppArmadillo` packages.

The `TDAvec` `R` package and a vignette showing its basic usage with examples are available on the CRAN repository[3]. For `Python` examples, we refer the readers to this page.

References

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., & Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, *18*(1), 218–252.

Adcock, A., Carlsson, E., & Carlsson, G. (2013). The ring of algebraic functions on persistence bar codes. *Homology, Homotopy and Applications*, *18*. https://doi.org/10.4310/HHA.2016.v18.n1.a21

[3]https://cran.r-project.org/web/packages/TDAvec/index.html

Ali, D., Asaad, A., Jimenez, M.-J., Nanda, V., Paluzo-Hidalgo, E., & Soriano-Trigueros, M. (2023). A survey of vectorization methods in topological data analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(12), 14069–14080.

Atienza, N., Gonzalez-Diaz, R., & Soriano-Trigueros, M. (2020). On the stability of persistent entropy and new summary functions for topological data analysis. *Pattern Recognition*, *107*, 107509. https://doi.org/10.1016/j.patcog.2020.107509

Awesome-TDA. (2024). https://github.com/FatemehTarashi/awesome-tda

Bauer, U. (2021). Ripser: Efficient computation of Vietoris-Rips persistence barcodes. *J. Appl. Comput. Topol.*, *5*(3), 391–423. https://doi.org/10.1007/s41468-021-00071-5

Brown, S., & Farivar, R. (2024). *TDApplied: Machine learning and inference for topological data analysis*. https://doi.org/10.32614/cran.package.tdapplied

Brunson, J. C., Demkowicz, B., & Choudhary, S. (2024). *Tdaunif: Uniform manifold samplers for topological data analysis*. https://doi.org/10.32614/CRAN.package.tdaunif

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, *16*(1), 77–102.

Carlsson, G. (2009). Topology and data. *Bulletin of the American Mathematical Society*, *46*(2). https://doi.org/10.1090/S0273-0979-09-01249-X

Chan, K. C., Islambekov, U., Luchinsky, A., & Sanders, R. (2022). A computationally efficient framework for vector representation of persistence diagrams. *Journal of Machine Learning Research*, *23*(268), 1–33.

Chazal, F., Fasy, B. T., Lecci, F., Rinaldo, A., & Wasserman, L. (2014). Stochastic convergence of persistence landscapes and silhouettes. *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, 474–483. https://doi.org/10.1145/2582112.2582128

Chazal, F., & Michel, B. (2021). An introduction to topological data analysis: Fundamental and practical aspects for data scientists. *Frontiers in Artificial Intelligence*, *4*. https://doi.org/10.3389/frai.2021.667963

Christopher, T., Nathaniel, S., & Rann, B. (2018). A lean persistent homology library for python. *Open J*, *925*.

Chung, Y.-M., & Lawson, A. (2022). Persistence curves: A canonical framework for summarizing persistence diagrams. *Advances in Computational Mathematics*, *48*(1), 1–42. https://doi.org/10.1007/s10444-021-09893-4

Di Fabio, B., & Ferri, M. (2015). Comparing persistence diagrams through complex vectors. *Image Analysis and Processing—ICIAP 2015: 18th International Conference, Genoa, Italy, September 7-11, 2015, Proceedings, Part i 18*, 294–305.

Edelsbrunner, H., & Harer, J. (2010). *Computational topology: An introduction*. American Mathematical Soc.

Fasy, B. T., Kim, J., Lecci, F., Maria, C., Millman, D. L., & Rouvreau., V. (2021). *TDA: Statistical tools for topological data analysis*. https://doi.org/10.32614/CRAN.package.TDA

Ferri, M., & Landi, C. (1999). Representing size functions by complex polynomials. *Proc. Math. Met. In Pattern Recognition*, *9*, 16–19.

Hensel, F., Moor, M., & Rieck, B. (2021). A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, *4*, 681108. https://doi.org/10.3389/frai.2021.681108

Kališnik, S. (2019). Tropical coordinates on the space of persistence barcodes. *Foundations of Computational Mathematics*, *19*(1), 101–129.

Morozov, D. (2018). *Dionysus 2 – library for computing persistent homology*. https://github.com/mrzv/dionysus

Nanda, V., & Sazdanovic, R. (2013). Simplicial models and topological inference in biological systems. In *Discrete and topological models in molecular biology* (pp. 109–141). Springer. https://doi.org/10.1007/978-3-642-40193-0_6

Padellini, T., & Palini, F. (2020). *kernelTDA: Statistical learning with kernel for persistence diagrams*. https://CRAN.R-project.org/package=kernelTDA

Pearson, P., Muellner, D., & Singh, G. (2015). *TDAmapper: Topological data analysis using mapper*. https://github.com/paultpearson/TDAmapper

Perea, J. A., Munch, E., & Khasawneh, F. A. (2023). Approximating continuous functions on persistence diagrams using template functions. *Foundations of Computational Mathematics*, *23*(4), 1215–1272.

Richardson, E., & Werman, M. (2014). Efficient classification using the euler characteristic. *Pattern Recognition Letters*, *49*, 99–106. https://doi.org/10.1016/j.patrec.2014.07.001

Rouvreau, V. (2020). *GUDHI user and reference manual*. GUDHI Editorial Board.

Saul, N. (2019). *Persim 0.3.7*. https://persim.scikit-tda.org/en/latest/

Tauzin, G., Lupo, U., Tunstall, L., Pérez, J. B., Caorsi, M., Medina-Mardones, A. M., Dassatti, A., & Hess, K. (2021). Giotto-tda:: A topological data analysis toolkit for machine learning and data exploration. *Journal of Machine Learning Research*, *22*(39), 1–6.

Van Veen, H. J., Saul, N., Eargle, D., & Mangham, S. W. (2019). Kepler mapper: A flexible python implementation of the mapper algorithm. *Journal of Open Source Software*, *4*(42), 1315.

Wadhwa, R. R., Williamson, D. F. K., Dhawan, A., & Scott, J. G. (2018). TDAstats: R pipeline for computing persistent homology in topological data analysis. *Journal of Open Source Software*, *3*(28), 860. https://doi.org/10.21105/joss.00860

Wadhwa, R., Piekenbrock, M., & Scott, J. (2020). *Ripserr: Calculate persistent homology with ripser-based engines*. https://doi.org/10.32614/CRAN.package.ripserr

Wickham, H. (2016). *ggplot2: Elegant graphics for data analysis*. Springer-Verlag New York. ISBN: 978-3-319-24277-4

You, K., & Yu, B. (2021). *TDAkit: Toolkit for topological data analysis*. https://doi.org/10.32614/cran.package.tdakit