The Journal of Open Source Software

Ripser.py: A Lean Persistent Homology Library for Python

Christopher Tralie[1], **Nathaniel Saul**[2], **and Rann Bar-On**[1]

1 Department of Mathematics, Duke University **2** Department of Mathematics and Statistics, Washington State University

DOI: 10.21105/joss.00925

Software

- Review
- Repository
- Archive

Submitted: 17 August 2018
Published: 13 September 2018

Summary

Topological data analysis (TDA) (Edelsbrunner & Harer, 2010), (Carlsson, 2009) is a field focused on understanding the shape and structure of data by computing topological descriptors that summarize features as connected components, loops, and voids. TDA has found wide applications across nonlinear time series analysis (Perea & Harer, 2015), computer vision (Perea & Carlsson, 2014), computational neuroscience (Giusti, Pastalkova, Curto, & Itskov, 2015), (Bendich, Marron, Miller, Pieloch, & Skwerer, 2016), computational biology (Iyer-Pascuzzi et al., 2010), (Wu et al., 2017), and materials science (Kramar, Goullet, Kondic, & Mischaikow, 2013), to name a few of the many areas of impact in recent years.

Persistent homology (Edelsbrunner & Harer, 2010) is the main workhorse of TDA, and it computes a data structure known as the persistence diagram to summarize the space of stable topological features. The most commonly used scheme for generating persistence diagrams is the Vietoris Rips filtration (VR) since it is easily defined for any point cloud. In its naive implementation, VR is prohibitively slow, but recently a C++ library known as Ripser (Bauer, 2017) has been devised to aggregate all known computational speedups of the VR filtration into one concise implementation. Because of the unprecedented speed of Ripser, it has created a large user base for both research and applications. However, the library as it stands is only a command line tool and as a result, multiple efforts have been made to wrap the C++ library for use in other languages, often via clunky system calls to the command line.

In this work, we develop an intuitive interface for VR filtrations with Ripser at its core via Cython. We have gone through extensive testing via continuous integration frameworks to ensure it works across all platforms and as a result, Ripser.py is currently as easy to setup as `pip install ripser`. We see this package as particularly useful for mathematicians with little programming experience who would like to use TDA as an entry point into data science, or conversely for researchers with little understanding of Algebraic Topology who would like to apply TDA to their problem domain. To aid this, we have created a large set of Jupyter notebooks to showcase some of the many applications that are possible with this library.

Library Details

Ripser.py supplies two interfaces: one lightweight and functional interface, as well as an object-oriented interface designed to fit within the Scikit-Learn transformer paradigm (Pedregosa et al., 2011). We have merged together multiple branches of the original Ripser library ("sparse-distance-matrix," "representative-cocycles") to expose some lesser

known but incredibly useful features hidden in Ripser. Below we detail some of the special features made easy with our library.

Sparse filtrations

Ripser.py can accomodate sparse distance matrices using the `scipy.sparse` library. In an accompanying notebooks, we demonstrate how Ripser.py can be used in conjunction with "sparse filtration" approximation algorithms (Cavanna, Jahanseir, & Sheehy, 2015) to add further computational speedups to Ripser. Additionally, sparse matrices allow us to easily define a "sublevelset filtration," or the "watershed algorithm," for quantifying critical points in grid data such as time series and images, and Ripser.py includes a helper function to make this easy for image data. One of the notebooks demonstrates how this can be used to identify cells in an image, for instance.

Coefficient Fields

Ripser.py naturally allows the specification of arbitrary field coefficients. Most applications that use TDA use binary coefficients in the VR filtration, and as a result, most existing TDA software can only handle binary coefficients. Using other coefficient fields $\mathbb{Z}/p\mathbb{Z}$ for any prime p, makes it possible to detect "twists" in point cloud data. We have included an example notebook that shows how this can differentiate a point cloud sampled from the boundary of a Moebius strip from a point cloud sampled from an ordinary, untwisted loop.

Applications of this surprisingly appears in some important real world scenarios such as periodic time series analysis (Perea & Harer, 2015) and image analysis (Perea & Carlsson, 2014). This feature of Ripser.py has been used in a pipeline to synthesize slow motion videos (Christopher J Tralie & Berger, 2018) and is currently being used to quantify periodicities in repetitive motions with children with autism spectrum disorder (Christopher J. Tralie, Matthew, & Sapiro, 2018).

Representative Cocycles

TDA is generally used to quantify topological features, but there has been some research on localizing topological features back in the point cloud. Ripser.py can return "representative cocycles" associated to different homology classes (topological features), for sparse and dense filtrations over any field. Use of this feature can be viewed as topological nonlinear dimension reduction (NLDR). Examples of this can be seen in mapping the point cloud to a circle (Silva, Morozov, & Vejdemo-Johansson, 2011), which is useful for parameterizing periodic data, or in mapping the point cloud to the projective plane (Perea, 2018), which shows up in analysis of image patches.

Higher Order Homology

Because of long standing speed concerns with TDA, most applications have focused on 0-dimensional (connected components) and 1-dimensional (loops) homology features from VR filtrations. However, Ripser.py is fast enough to compute 2-dimensional homology (voids e.g. empty space in a basketball) for modest point clouds. This feature has been used to quantify quasiperiodic phenomena in videos of patients with vocal fold disorders (Christopher J. Tralie & Perea, 2018), and we anticipate more researchers will use the feature for other problems now that it is computationally accessible.

Source Code

The source code for Ripser.py is available on Github through the Scikit-TDA organization https://github.com/scikit-tda/Ripser.py. The original Ripser library can be found at https://github.com/Ripser/Ripser/

Acknowledgements

Christoher Tralie and Rann Bar-On were supported by an NSF big data grant DKA-1447491. Nathaniel Saul was partially supported by NSF DBI-1661348 and by Washington NASA Space Grant Consortium, NASA Grant #NNX15AJ98H. We thank Ulrich Bauer for the original Ripser library and for valuable feedback during development of Ripser.py. We also thank Jose Perea, William Guss, and Matija Čufar for helpful feedback and bug fixes. Finally, we thank the students of the "Topological Data Analysis and Persistent Homology" workshop in Levico, Italy for beta testing the code.

References

Bauer, U. (2017). Ripser: A lean c++ code for the computation of vietoris-rips persistence barcodes. *Software available at https://github.com/Ripser/ripser*.

Bendich, P., Marron, J., Miller, E., Pieloch, A., & Skwerer, S. (2016). Persistent homology analysis of brain artery trees. *Annals of Applied Statistics*, *10*(1), 198–218. doi:10.1214/15-AOAS886

Carlsson, G. (2009). Topology and data. *Bulletin of the American Mathematical Society*, *46*(2), 255–308.

Cavanna, N. J., Jahanseir, M., & Sheehy, D. R. (2015). A geometric perspective on sparse filtrations. *Proceedings of the Canadian Conference in Computational Geometry*.

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

Giusti, C., Pastalkova, E., Curto, C., & Itskov, V. (2015). Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, *112*(44), 13455–13460. doi:10.1073/pnas.1506407112

Iyer-Pascuzzi, A. S., Symonova, O., Mileyko, Y., Hao, Y., Belcher, H., Harer, J., Weitz, J. S., et al. (2010). Imaging and analysis platform for automatic phenotyping and trait ranking of plant root systems. *Plant Physiology*, *152*(3), 1148–1157. doi:10.1104/pp.109.150748

Kramar, M., Goullet, A., Kondic, L., & Mischaikow, K. (2013). Persistence of force networks in compressed granular media. *Phys. Rev. E*, *87*(4), 042207. doi:10.1103/PhysRevE.87.042207

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perea, J. A. (2018). Multiscale projective coordinates via persistent cohomology of sparse filtrations. *Discrete & Computational Geometry*, *59*(1), 175–225. doi:10.1007/s00454-017-9927-2

Perea, J. A., & Carlsson, G. (2014). A klein-bottle-based dictionary for texture representation. *International Journal of Computer Vision*, *107*(1), 75–97. doi:10.1007/s11263-013-0676-2

Perea, J. A., & Harer, J. (2015). Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, *15*(3), 799–838. doi:10.1007/s10208-014-9206-z

Silva, V. de, Morozov, D., & Vejdemo-Johansson, M. (2011). Persistent cohomology and circular coordinates. *Discrete & Computational Geometry*, *45*(4), 737–759. doi:10.1007/s00454-011-9344-x

Tralie, C. J., & Berger, M. (2018). Topological eulerian synthesis of slow motion periodic videos. In *IEEE international conference on image processing*.

Tralie, C. J., & Perea, J. A. (2018). (Quasi)Periodicity quantification in video data, using topology. *SIAM Journal on Imaging Sciences*, *11*(2), 1049–1077. doi:10.1137/17M1150736

Tralie, C. J., Matthew, G. S., & Sapiro, G. (2018). Automated detection of stereotypical motor movements in children with autism spectrum disorder using geometric feature fusion. *International Society for Autism Research (INSAR)*.

Wu, P., Chen, C., Wang, Y., Zhang, S., Yuan, C., Qian, Z., Metaxas, D., et al. (2017). Optimal topological cycles and their application in cardiac trabeculae restoration. In M. Niethammer, M. Styner, S. Aylward, H. Zhu, I. Oguz, P.-T. Yap, & D. Shen (Eds.), *Information processing in medical imaging* (pp. 80–92). Cham: Springer International Publishing.