Ripserer.jl: flexible and efficient persistent homology computation in Julia

Matija Čufar[1]

1 Independent Researcher

DOI: 10.21105/joss.02614

Software

- Review
- Repository
- Archive

Editor: Viviane Pons

Reviewers:

- @juliohm
- @sauln

Submitted: 26 August 2020
Published: 19 October 2020

Introduction

Persistent homology (Edelsbrunner & Harer, 2008) is a relatively recent computational technique that extracts topological information from various kinds of datasets. This topological information gives us a good overview of the global shape of the data as well as giving us a description of its local geometry. Since its introduction, it has been used in a diverse range of applications, including biology (Bernoff & Topaz, 2016), material science (Lee et al., 2017), signal processing (Tralie, 2016), and computer vision (Asaad & Jassim, 2017). A problem persistent homology faces is the very large size of combinatorial structures it has to work with. Recent algorithmic advances employ various computational shortcuts to overcome this problem.

Among the most successful implementations of persistent homology is Ripser (Bauer, 2019). With its speed and low memory usage, it makes persistent homology practical for larger datasets, even in higher dimensions. The introduction of Ripser has spawned a whole cottage industry of extensions and wrappers. Some examples include Ripser++ (Zhang, Xiao, & Wang, 2020), Lock-free Ripser (Morozov & Nigmetov, 2020), Ripser.py (Tralie, Saul, & Bar-On, 2018), Cubical Ripser (Kaji, Sudo, & Ahara, 2020), and Flagser (Lütgehetmann, Govc, Smith, & Levi, 2020).

In the Julia (Bezanson, Edelman, Karpinski, & Shah, 2017) space, there are few persistent homology packages available. The ones we were able to find include Eirene.jl[1] (Henselman & Ghrist, 2016), ComputationalHomology.jl[2], Sparips.jl[3] (Brehm & Hardering, 2018), and PersistentCohomology.jl[4], of which only Eirene.jl is available through the Julia package manager.

Statement of Need

A significant hurdle in developing new approaches to persistent homology stems from the fact that developing an efficient implementation of its matrix reduction algorithm is nontrivial.

To solve this problem, we introduce Ripserer.jl, a pure Julia implementation of persistent homology based on the algorithm that powers Ripser. It provides users with an intuitive user interface and is readily useful as a topological data analysis framework. The other main feature Ripserer.jl provides is the ability to hook into its algorithm through an API. This allows researchers to experiment with different approaches to persistent homology without having to reimplement the algorithm from scratch or forking an existing repository.

[1]https://github.com/Eetion/Eirene.jl
[2]https://github.com/wildart/ComputationalHomology.jl
[3]https://github.com/bbrehm/Sparips.jl
[4]https://github.com/piever/PersistentCohomology.jl

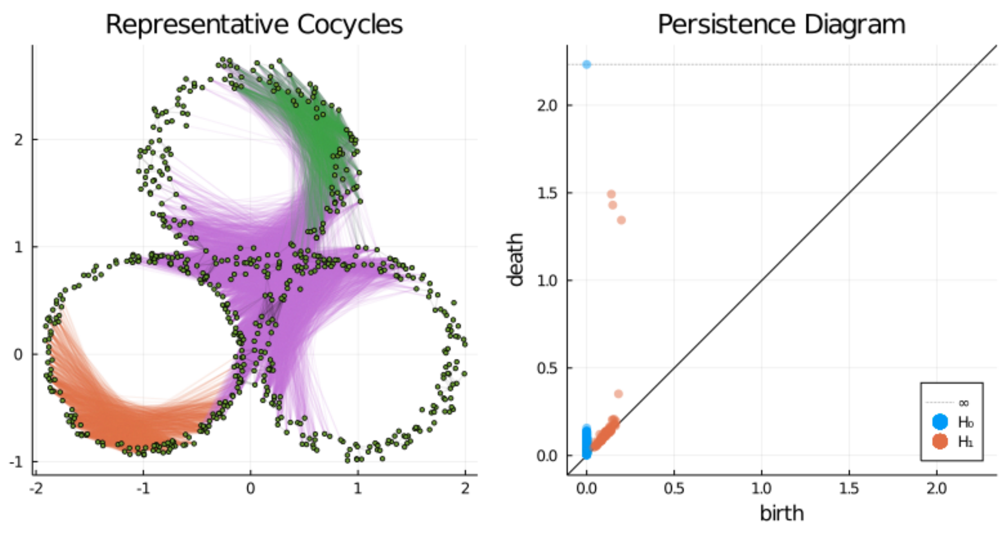

Figure 1: Example visualizations. The plot on the left shows the three main representative cocycles in the data. The right plot shows the persistence diagram.

Summary

Along with its companion package, PersistenceDiagrams.jl[5], Ripserer.jl provides a featureful environment for computing persistent homology and integrating it with the rest of Julia's data science stack. At the time of writing, it offers the following features.

- Fast Vietoris-Rips, alpha complex, and cubical persistent homology computation.
- Representative cocycle and critical simplex computation.
- Support for coefficients in any, possibly user-defined, field.
- Convenient persistence diagram and representative cocycle visualization via Plots.jl[6] recipes.
- Bottleneck and Wasserstein matching and distance computation.
- Various persistence diagram vectorization functions, implemented with persistence images (Adams et al., 2017) and persistence curves (Chung & Lawson, 2019).
- Easy extensibility through a documented API.

Our benchmarks[7] show that Ripserer's performance is very close to that of Ripser. It tends to be slightly slower for dense inputs and slightly faster for very sparse inputs. In the cubical case, we compared it to Cubical Ripser. There the performance was worse, taking up to 3 times as long to compute some results. This is expected as Cubical Ripser is much more specialized for its use case and even splits its code into different repositories for different dimensions.

We have not compared performance with newer, parallel implementations such as Ripser++ or lock-free Ripser. Judging from the benchmarks they provide, we expect them to perform much better. Their downside, however, is that they require powerful hardware, such as GPUs or large numbers of processors.

[5]https://github.com/mtsch/PersistenceDiagrams.jl
[6]https://github.com/JuliaPlots/Plots.jl
[7]https://mtsch.github.io/Ripserer.jl/dev/benchmarks/

Acknowledgments

We would like to thank Žiga Virk for comments, suggestions and ideas, and Ulrich Bauer for making the source code of Ripser freely available.

References

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., et al. (2017). Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, *18*(1), 218–252. Retrieved from https://dl.acm.org/doi/10.5555/3122009.3122017

Asaad, A., & Jassim, S. (2017). Topological data analysis for image tampering detection. In *International workshop on digital watermarking* (pp. 136–146). Springer. doi:10.1007/978-3-319-64185-0_11

Bauer, U. (2019). Ripser: Efficient computation of vietoris-rips persistence barcodes. *arXiv preprint arXiv:1908.02518*. Retrieved from https://arxiv.org/abs/1908.02518

Bernoff, A. J., & Topaz, C. M. (2016). Biological aggregation driven by social and environmental factors: A nonlocal model and its degenerate cahn–hilliard approximation. *SIAM Journal on Applied Dynamical Systems*, *15*(3), 1528–1562. doi:10.1137/15M1031151

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM review*, *59*(1), 65–98. doi:10.1137/141000671

Brehm, B., & Hardering, H. (2018). Sparips. *arXiv preprint arXiv:1807.09982*. Retrieved from https://arxiv.org/abs/1807.09982

Chung, Y.-M., & Lawson, A. (2019). Persistence curves: A canonical framework for summarizing persistence diagrams. *arXiv preprint arXiv:1904.07768*. Retrieved from https://arxiv.org/abs/1904.07768

Edelsbrunner, H., & Harer, J. (2008). Persistent homology-a survey. *Contemporary mathematics*, *453*, 257–282. doi:10.1090/conm/453/08802

Henselman, G., & Ghrist, R. (2016). Matroid filtrations and computational persistent homology. *arXiv preprint arXiv:1606.00199*. Retrieved from https://arxiv.org/abs/1606.00199

Kaji, S., Sudo, T., & Ahara, K. (2020). Cubical ripser: Software for computing persistent homology of image and volume data. *arXiv preprint arXiv:2005.12692*. Retrieved from https://arxiv.org/pdf/2005.12692.pdf

Lee, Y., Barthel, S. D., Dłotko, P., Moosavi, S. M., Hess, K., & Smit, B. (2017). Quantifying similarity of pore-geometry in nanoporous materials. *Nature communications*, *8*, 15396. doi:10.1038/ncomms15396

Lütgehetmann, D., Govc, D., Smith, J. P., & Levi, R. (2020). Computing persistent homology of directed flag complexes. *Algorithms*, *13*(1), 19. doi:10.3390/a13010019

Morozov, D., & Nigmetov, A. (2020). Towards lockfree persistent homology. In *Proceedings of the 32nd acm symposium on parallelism in algorithms and architectures* (pp. 555–557). doi:10.1145/3350755.3400244

Tralie, C. (2016). High-dimensional geometry of sliding window embeddings of periodic videos. In *32nd international symposium on computational geometry (socg 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2016.71

Tralie, C., Saul, N., & Bar-On, R. (2018). Ripser. Py: A lean persistent homology library for python. *Journal of Open Source Software*, *3*(29), 925. doi:10.21105/joss.00925

Zhang, S., Xiao, M., & Wang, H. (2020). GPU-accelerated computation of vietoris-rips persistence barcodes. *arXiv preprint arXiv:2003.07989*. Retrieved from https://arxiv.org/abs/2003.07989