LUMÁWIG: Un-bottling the bottleneck distance for zero dimensional persistence diagrams at scale

Paul Samuel P. Ignacio[1*], Jay-Anne B. Bulauan[1], David Uminsky[2]

[1]University of the Philippines Baguio, Baguio City, Philippines 2600, [2]University of Chicago, Chicago, Il, United States 60637

Introduction

A central task in TDA is the computation of the bottleneck distance between two persistence diagrams. For diagrams induced by the Rips filtration, all dimension zero signatures are born at the start of the filtration, capturing cluster merging dynamics akin to that observed by hierarchical clustering methods. This is illustrated in Figure 1.

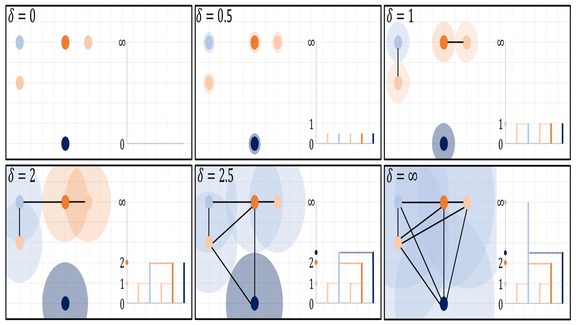

Figure 1:A Rips filtration produces dimension zero persistence diagrams whose birth times are all zero and death times correspond to the merge heights in a dendrogram.

The first, and for a long time the only, publicly available implementation of the bottleneck distance for persistence diagrams is in the library DIONYSUS, released by Morozov [1] in 2010. In 2017, Morozov et al. [2] provided an improved implementation in the library HERA by exploiting geometry. We take inspiration from this idea of exploiting the geometry of persistence diagrams to extract computational speed-up. By considering dimension 0 persistence diagrams induced from the Rips filtration, we can approach the problem via a different framework, birthing a new efficient algorithm for computing the bottleneck distance.

Bypassing Matchings

To bypass the overwhelming matching step, the key idea is to begin with a specific initial bijection that one can methodically modify to optimize the norm between matched points. Let X and Y be two 0-dimensional persistence diagrams whose death times are arranged from largest to smallest. Let N be the length of X and define $Z = [z_i]_1^{|Y|}$ where

$$z_i = \begin{cases} |x_i - y_i| & \text{if } i \leq N \\ y_i/2 & \text{otherwise} \end{cases}$$

and $l = \arg\max(Z)$. Then the following hold.

Lemma 1

If $N < |Y|$ and $\max(Z) \leq y_{N+1}/2$, then $d_B(X,Y) = y_{N+1}/2$ where y_{N+1} is the largest death time of a point in Y matched to the diagonal.

Lemma 2

Let ζ be the second largest entry of Z.

1. If $\max(Z) \leq \max(x_l, y_l)/2$, then
$$d_B(X,Y) = \max(Z).$$
2. If $\zeta < \max(x_l, y_l)/2 < \max(Z)$, then
$$d_B(X,Y) = \max(x_l, y_l)/2.$$
3. If $\zeta \geq \max(x_l, y_l)/2$ and $m \geq l$ for every m such that $z_m \geq \max(x_l, y_l)/2$, then
$$d_B(X,Y) = \max(x_l, y_l)/2.$$
4. If $\zeta \geq \max(x_l, y_l)/2$ and there exists $m < l$ such that $z_m \geq \max(x_l, y_l)/2$, then there exists a bijection τ between X and Y such that one of the three preceding cases holds and where
$$\max ||x - \tau(x)||_\infty < \max ||x - \phi(x)||_\infty.$$

Benchmarking

We benchmark LUMÁWIG against the current state-of-the-art implementation of the bottleneck distance in HERA. We simulate 100 0-dimensional persistence diagrams, pair each diagram with another simulated diagram with as much as 80% more or fewer points, then compute the bottleneck distance between the pair. The running time distributions for 100 computation of bottleneck distance on increasing diagram sizes (from 1,000 to 30,000 points) are plotted in Figure 2.

Empirical Tests For Complexity

We examine LUMÁWIG's running time in the computation of dimension zero bottleneck distance in four settings. We vary the size of the diagrams and the range of values the death times are drawn from. The four settings are (i) equal size and range; (ii) equal size but different range; (iii) different size but equal range; and (iv) different size and range. We increase the diagram size from 1,000 points to 1 million points for the first two settings. However, due to increased computational time, we increase to only 100,000 points in the third setting, and to 400,000 points in the fourth setting. In all settings we perform 100 computations.

Running time of Lumáwig vs Hera.

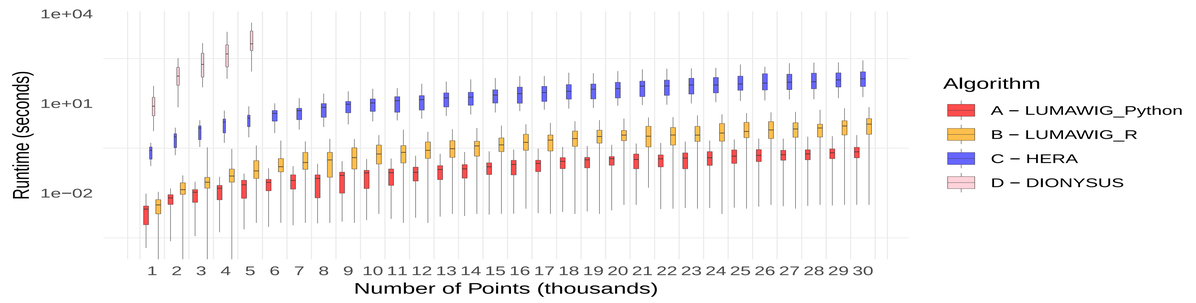

Figure 2:Running time (seconds in log scale) of LUMÁWIG versus the current state-of-the-art implementation in HERA. Five boxplots for the running time of the original algorithm in DIONYSUS are superimposed for reference.

Figure 3:Median running time in the computation of bottleneck distance between two diagrams with varying size and range settings fitted with regression curves. The plots are stacked from top to bottom based respectively on the four setting defined above.

Conclusion

LUMÁWIG outperforms by several orders of magnitude, the state-of-the art implementation of dimension zero bottleneck distance in terms of running time. In [3], we show that LUMÁWIG also recovers the exact bottleneck distance produced by DIONYSUS. As LUMÁWIG generally enjoys linear complexity as shown by our empirical tests, we are able to present in this work the first instance, to the best of our knowledge, that the bottleneck distance is used in practice on data of magnitude and scale in the order of up to a million. This opens the opportunity to scale TDA to data sets of sizes encountered in machine learning and utilize persistence diagrams in a manner that goes beyond the simple use of the most persistent components. We believe that LUMÁWIG is a significant contribution in this direction as it affords a viable tool to process and utilize dimension zero persistence diagrams in comparing evolving connectivity information between larger data sets.

References

[1] Morozov, D., DIONYSUS library for computing persistent homology. mrzv.org/software/dionysus.

[2] Kerber, M., Morozov, D., Nigmetov, A., Geometry Helps to Compare Persistence Diagrams, J. Exp. Algorithmicsm 2017, 22(1), Article 1.4, 20 pages.

[3] Ignacio, P.S., Bulauan, J., Uminsky, D., Lumáwig: An Efficient Algorithm for Dimension Zero Bottleneck Distance Computation in Topological Data Analysis, Algorithms, (2020), 13(11), 291.

Acknowledgements

We thank the University of the Philippines Baguio for supporting our participation in this conference.

*Correspondence: ppignacio@up.edu.ph