We may often observe that distinct shapes and structure emerge from the locations of individuals or discrete data samples. *Topological data analysis* (TDA) aims to summarize these patterns using concepts from algebraic topology and computational geometry. In this blog post, we will focus on the ideas and implementation of *persistent homology*, which is the most commonly-used form of TDA.

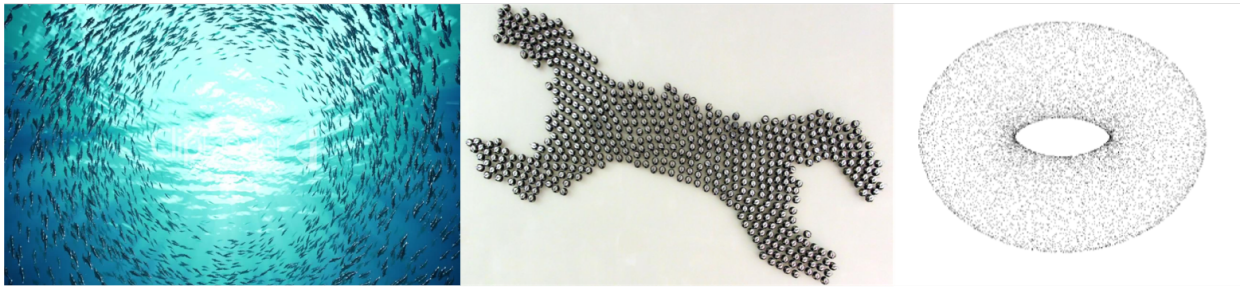

Figure 1: Schools of fish, swarming robots, and collections of data points can exhibit emergent shape or structure.

Homology

To discuss persistent homology, we must first understand *homology*, which summarizes the shape of topological objects. An object's homology is often summarized through the *Betti numbers*, which count the number of k-dimensional holes in the object. For example, the zeroth Betti number, β_0, counts the number of connected components in an object, the first Betti number, β_1, counts the number of loops, the second Betti number, β_2, counts the number of trapped volumes, etc.

Figure 2: The homology for a two-dimensional circle, the surface of a soccer ball, and the surface of a torus.

The Betti numbers for three topological objects are provided in Figure 2. A two-dimensional circle has one connected component, one loop, and no trapped volumes. The

surface of a soccer ball has one connected component, no loops, and one trapped volume. To understand why there are no loops on the soccer ball, imagine tying a taught rubber band around it's surface. If you moved the rubber band along the surface (while keeping it taught), the rubber band will eventually fall off, so there is no loop. The surface of a torus has one connected component, two loops, and one trapped area. There are two loops on the torus because if you placed a rubber band along either of the red curves in Figure 2, you would not be able to remove the rubber band without tearing it.

Simplicial complexes

The homology of the following collection of data points is simple – β_0 is equal to the number of data points and all other Betti numbers are zero!

However, we may believe that there is an underlying structure to these discrete data points. To model this structure, we construct a *simplicial complex*, or collection of simple shapes that connect the data points together. We include each data point in our simplicial complex. We then choose a threshold distance, ϵ, and include the line connecting any two data points that are within ϵ of each other to the simplicial complex, as shown in Figure 3. Additionally, we will place a triangle connecting any three datapoints that are all within ϵ of each other. We could go on further (for example, by placing a tetrahedron between any four points that are each pairwise within ϵ of each other), but we'll stop here. Note that we have described how to create the Vietoris-Rips complex in this example, though there are other ways to construct simplicial complexes. For the chosen value of ϵ in Figure 3, we notice that the simplicial complex's Betti numbers are $\beta_0 = 3, \beta_1 = 1, \beta_2 = 0$, etc.

Persistent homology

In the previous section, we computed the homology of the data by constructing a simplicial complex, K_ϵ, by connecting points that are pairwise within ϵ of each other. In Persistent homology, we consider a range of increasing epsilon values, $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_n$ and create a filtration of simplicial complexex for each value of ϵ_i: $K = \{K_{\epsilon_i}\}_{i=1}^n$. We note that

$$K_{\epsilon_1} \subseteq K_{\epsilon_2} \subseteq \cdots \subseteq K_{\epsilon_n}. \tag{1}$$

To compute the persistent homology of a data set, we track the values of ϵ over which topological features are born and die. In Figure 4, we notice each data point is it's own connected component for $\epsilon = 0.03$. We say that each of these connected components are

Figure 3: (Left)The Vietoris Rips complex for a collection of points is constructed by joining together data points that are all pairwise within some threshold value ϵ of each other. (Right) The resulting simplicial complex is made up of dots, lines, and filled in triangles.

born at $\epsilon = 0$. At $\epsilon = 0.04$, however, we notice that two of our data points are now connected, so one of our connected components has "died" before $\epsilon = 0.04$. At $\epsilon = 0.09$, we notice that a loop has been born in our simplicial complex, but it becomes filled in and dies by $\epsilon = 0.11$.

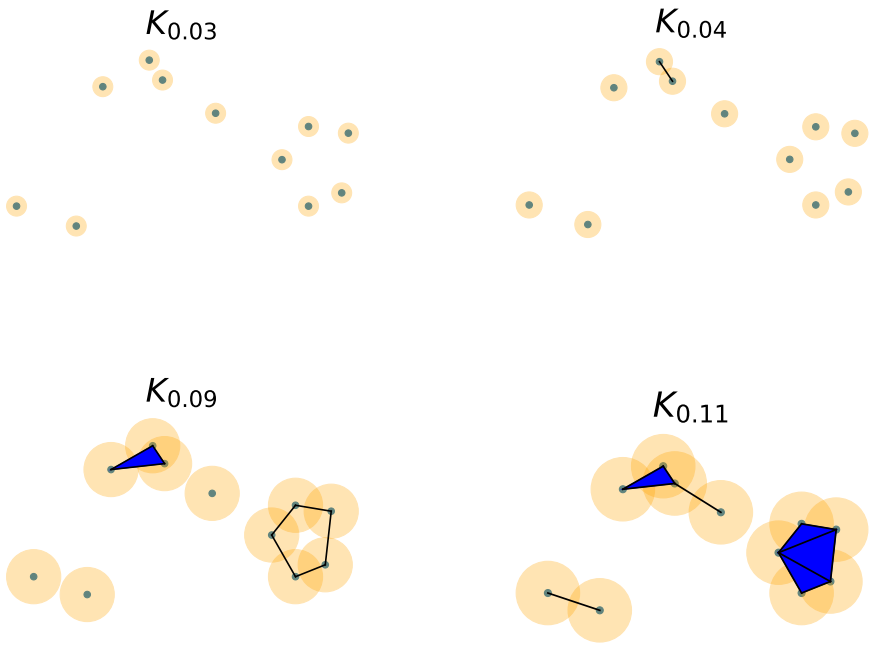

Figure 4: The Vietoris Rips complex for a collection of points is constructed by joining together data points that are all pairwise within some threshold value ϵ of each other.

We represent the birth and death of each topological feature using persistence diagrams (the x-value of each point depicts when a feature was born and the y-values depicts when it died). For example, there is an orange dot at (birth, death) = (0.0905, 0.1077), because there is a loop born at $\epsilon = 0.0905$ that becomes filled in at $\epsilon = 0.1077$.

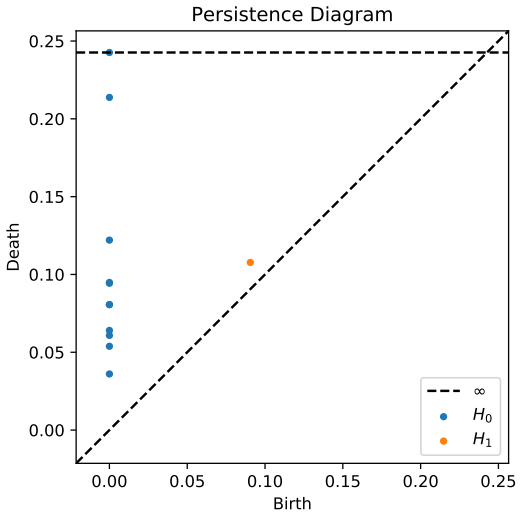

Figure 5: Persistence diagrams summarize the birth and death values for each topological feature in a given data set. Blue dots denote connected components and orange dots denote loops.

TDA and mathematical modeling

While we used TDA to summarize a simple collection of points in the previous section, we can also use TDA to analyze complex patterns that result from mathematical model simulations [1]. Consider the agent-based model from [2]: we have N interacting particles with two-dimensional positions, $\boldsymbol{x}_i(t) = (x_i(t), y_i(t)), i = 1, \ldots, N$, and velocities, $\boldsymbol{v}_i(t) = (v_{x,i}(t), v_{y,i}(t))$, that change over time. Using Newton's Law, we can determine a differential equation model for the motion of each particle's position and velocity as follows:

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} &= \boldsymbol{v}_i \\ m\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} &= (\alpha - \beta|\boldsymbol{v}_i|^2)\boldsymbol{v}_i - \nabla_i U(\boldsymbol{x_i}), i = 1, \ldots, N \\ U(\boldsymbol{x}_i) &= \sum_{j \neq i} \left[C_r e^{-|\boldsymbol{x}_i - \boldsymbol{x}_j|/\ell_r} - C_a e^{-|\boldsymbol{x}_i - \boldsymbol{x}_j|/\ell_a} \right]. \end{aligned} \tag{2}$$

Equation (2) describes how forces of self propulsion (with parameter α), friction (with parameter β), as well as attraction (with magnitude C_a over characteristic length ℓ_a) and repulsion (with magnitude C_r over characteristic length ℓ_r) between agents affect particle

motion. When $\alpha = 1.5, \beta = 0.5, C_a = 1.0, C_r = 0.1, \ell_a = 1.0, \ell_r = 0.1$, the model outputs a double ring pattern comprised of agents moving clockwise and counterclockwise (Fig. 6) [1].

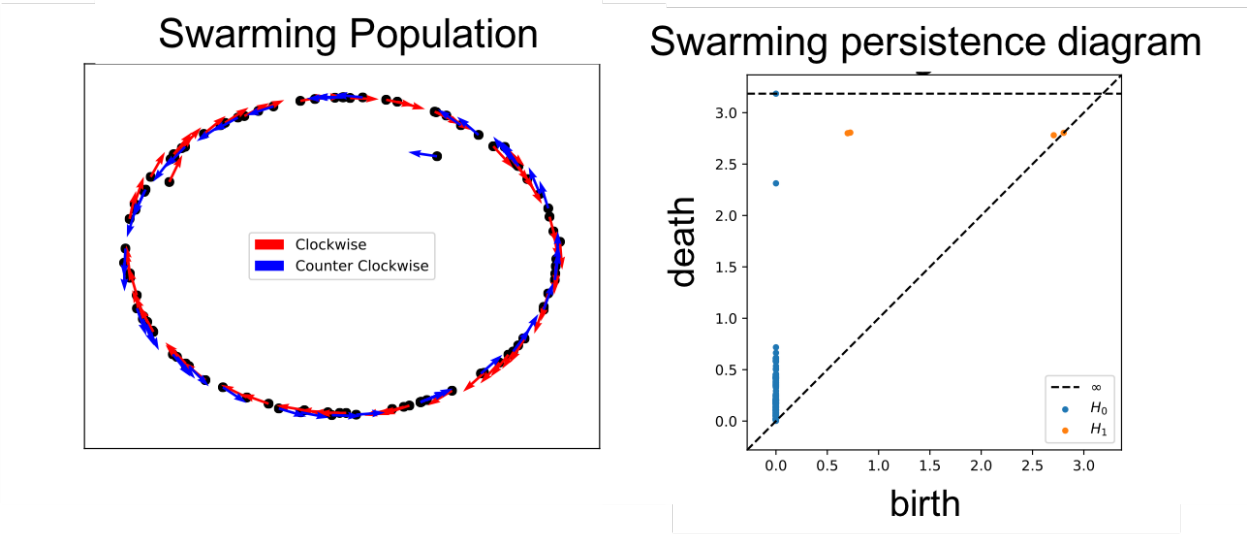

Figure 6: TDA computation of an agent-based model of biological swarming. (Left) A simulation of the D'Orsogna Model exhibits agents moving in both clockwise (red) and counter clockwise (blue) orientations. (Right) The persistence diagram of this simulation snapshot shows the presence of two loops that are born close to $\epsilon = 0.7$ and die near $\epsilon = 2.8$.

Persistent homology provides an informative way to summarize this observed patterning. We can create a four-dimensional point cloud comprised of the position and velocity information for each agent. We then compute the persistence diagram of the model snapshot from Figure 6 using this four dimensional data. We observe that two loops that are born near $\epsilon = 0.7$ and die near $\epsilon = 2.8$. These two loops likely correspond to the two rings moving clockwise or counter clockwise in the model simulation. Other model parameters will lead to different shapes and persistence diagrams; we encourage you to play around with the parameter values provided in Table 1 using the code provided at [3]!

	α	β	C_a	C_r	ℓ_a	ℓ_r
Parameter set 1	1.5	0.5	1.0	0.1	1.0	0.1
Parameter set 2	1.5	0.5	1.0	0.5	1.0	0.1
Parameter set 3	1.5	0.5	1.0	0.9	1.0	0.5
Parameter set 4	1.5	0.5	1.0	0.1	1.0	0.5
Parameter set 5	1.5	0.5	1.0	2.0	1.0	0.9

Table 1: Table of parameter values to test out using the code provided at [3].

TDA computation

The python code used to generate the figures in this this tutorial is available at [3]. We use the Ripser python package to compute the persistent homology of the dataset because it

has been found to be the best performing algorithm for computing the Vietoris-Rips complex [4, 6]. To understand to use linear algebra to compute the homology for a simplicial complex, we direct the interested reader to [5].

References

[1] D. Bhaskar, A. Manhart, J. Milzman, J. T. Nardini, K. M. Storey, C. M. Topaz, and L. Ziegelmeier. Analyzing collective motion with machine learning and topology. *Chaos*, 29(12):123125, Dec. 2019. Publisher: American Institute of Physics.

[2] M. R. D'Orsogna, Y. L. Chuang, A. L. Bertozzi, and L. S. Chayes. Self-Propelled Particles with Soft-Core Interactions: Patterns, Stability, and Collapse. *Phys. Rev. Lett.*, 96(10):104302, Mar. 2006.

[3] J. Nardini. Tda_tutorial. https://github.com/johnnardini/TDA_tutorial.

[4] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington. A roadmap for the computation of persistent homology. *European Physical Journal – Data Science*, 6(17):1–38, 2017.

[5] C. Topaz. Chad's self-help homology tutorial for the simple(x) minded.

[6] C. Tralie, N. Saul, and R. Bar-On. Ripser.py: A lean persistent homology library for python. *The Journal of Open Source Software*, 3(29):925, Sep 2018.