Deforming Point Clouds to Control Persistent Homology

Anja Laharnar[1*], **Vanja Stojanović**[1†]

[1] *Faculty of Mathematics and Physics, University of Ljubljana, Slovenia*
E-mail: *a193875@student.uni-lj.si, †vs66277@student.uni-lj.si

January 2026

Abstract

This work addresses the inverse problem in persistent homology: given a target topological signature, how must a point cloud configuration evolve to match it? We propose an optimization framework where point coordinates are treated as trainable parameters and updated via gradient descent. By utilizing the differentiability of the Vietoris-Rips filtration, we define a loss function that simultaneously maximizes the persistence of target homological features and suppresses topological noise. We explicitly formulate the gradients for critical birth and death values and demonstrate the algorithm's effectiveness in deforming 2D point clouds to recover clear topological signals. Finally, we provide a computational complexity analysis of the proposed optimization step.

1 Introduction

Topological Data Analysis (TDA) is a subset of topological concepts and methods used to extract structural information from complex, high-dimensional datasets. Its core tool, persistent homology, provides a multiscale summary of topological features, such as connected components (H_0), loops (H_1), and voids (H_2), which is used for classifying topological features.

While computing the topological signature, or persistence diagram, of a given point cloud is well-understood and computationally efficient, the inverse problem remains a significant challenge. The inverse problem asks: given a desired topological signature, can we generate or deform a point cloud to match it?

Solving this problem allows for topological control over data. For instance, in machine learning, topological priors are used to regularize decision boundaries,

ensuring classifiers do not overfit to noise [2]. In deep generative models, differentiable topology layers force generated images to maintain structural coherence (e.g., ensuring disconnected lines in a digit are joined) [3]. Furthermore, in material science, the inverse design of porous materials with specific void distributions is essential for optimizing physical properties like permeability and elasticity [6].

In this work, we address the problem of Feature Manipulation via topological optimization. We treat point cloud positions as trainable parameters and optimize them via gradient descent. Our method focuses on the precise control of individual homological features. We then use this algorithm to deform the persistence diagram of one point cloud into that of another.

All of the code, plots, and animations, are available in a public GitHub repository[1].

1.1 Preliminaries and Notation

We consider a point cloud $P = \{x_1, \ldots, x_N\} \subset \mathbb{R}^d$. To analyze its topology, we construct the Vietoris-Rips filtration, denoted $\mathcal{R}(P)$. A simplex σ is included in the complex at scale ϵ if all pairwise distances between its vertices are at most ϵ.

The evolution of homology groups across scales is summarized in a Persistence Diagram $D(P)$, consisting of birth-death pairs (b_k, d_k). A key property of the Rips filtration is that the birth and death values are determined entirely by the lengths of specific *critical edges* in the underlying graph. We denote the *persistence* (or lifespan) of a feature as $\text{pers}(k) = d_k - b_k$. Features with low persistence are typically regarded as sampling noise, while those with high persistence represent significant structural properties.

1.2 Related Work

The feasibility of topological optimization rests on the differentiability of the persistence map. Turner et al. [7] and Gameiro et al. [4] laid the theoretical groundwork, showing that the map from point coordinates to persistence diagrams is continuous and differentiable almost everywhere.

Recent advancements have focused on integrating these gradients into optimization frameworks. Carrière et al. [1] provided a general framework for optimizing topological functionals, enabling the use of persistence-based losses in neural networks. Similarly, Gabrielsson et al. [3] introduced a "topology layer" for deep learning that allows for end-to-end training with topological constraints.

Our work builds directly on the geometric formulation by Hiraoka et al. [5] and Obayashi et al. [6], who explicitly derive the gradient of critical values with respect

[1] `https://github.com/Lavennie/RT2-Topological-Optimization`

to vertex positions. While previous works often focus on global statistics (e.g., minimizing total persistence entropy), our algorithm specifically targets the *feature manipulation* task: selectively maximizing the lifespan of a single target feature h^* while crushing the background noise, effectively "denoising" the topology of the point cloud.

2 Optimizing the Birth and Death Process

The core objective of our study is to solve the inverse problem in persistent homology: given a target topological signature, how must the point cloud configuration $P = \{x_1, ..., x_n\} \subset \mathbb{R}^2$ evolve? We approach this as an optimization problem where the coordinates of the point cloud are updated via gradient descent to minimize a topological loss function $\mathcal{L}_{topo}$.

We utilize the Vietoris-Rips filtration, denoted $\mathcal{R}(P)$. A simplex σ is included in the complex if all pairwise distances between its vertices are at most ϵ. Consequently, the filtration value (the time of appearance) of any simplex σ is determined entirely by the length of its longest edge.

2.1 Differentiability of Critical Values

The persistence diagram $D(P)$ consists of birth-death pairs (b_k, d_k) for each homological feature k. The map from point coordinates to these persistence values is continuous and piecewise differentiable. As established in the literature on topological optimization [4, 1], the gradient is well-defined almost everywhere (specifically, whenever the point cloud is in a generic configuration).

The optimization relies on the fact that every birth b and death d in a Rips complex corresponds to the length of a specific *critical edge* connecting two vertices in P. We formalize the gradient of these values using the formulation presented in recent work on topological representation learning [5].

Lemma 2.1 (Gradient of Critical Values [5])**.** *Let v be a critical value (either birth or death) in the persistence diagram of the Rips complex $\mathcal{R}(P)$. Assume v is determined by the critical edge e_{ij} connecting vertices x_i and x_j in P, such that $v = \|x_i - x_j\|$. The gradient of v with respect to a point $x_k \in P$ is given by:*

$$\nabla_{x_k} v = \begin{cases} \frac{x_k - x_j}{\|x_k - x_j\|} & ; \text{if } k = i \\ \frac{x_k - x_i}{\|x_k - x_i\|} & ; \text{if } k = j \\ 0 & ; \text{otherwise} \end{cases} \tag{1}$$

The lemma defines the gradient direction for increasing the length of a critical edge. However, to maximize the persistence of a feature (defined as $d-b$), we must

apply opposite forces depending on the edge type: we aim to increase the death value d (pushing vertices apart, in the direction of $+\nabla d$) and decrease the birth value b (pulling vertices together, in the direction of $-\nabla b$). This sign inversion is naturally captured by the loss function derivatives.

2.2 Feature Manipulation

For the task of feature manipulation, we aim to prolong a specific 1-dimensional loop (the target feature h^*) while suppressing all other features (noise). Let (b^*, d^*) be the birth and death of the target, and $\{(b_i, d_i)\}_{i \in \text{noise}}$ be the persistence pairs of the noise features. We define the loss function for the feature manipulation (shortened as FM):

$$\mathcal{L}_{\text{FM}}(P) = -\lambda_{\text{target}}(d^* - b^*) + \lambda_{\text{noise}} \sum_{i \in \text{noise}} (d_i - b_i)^2 \tag{2}$$

Using the chain rule and Lemma 2.1, the update rule for a point x_k at step t is:

$$x_k^{(t+1)} = x_k^{(t)} - \eta \left(-\lambda_{\text{target}}(\nabla_{x_k} d^* - \nabla_{x_k} b^*) + \lambda_{\text{noise}} \sum_{i \in \text{noise}} 2(d_i - b_i)(\nabla_{x_k} d_i - \nabla_{x_k} b_i) \right) \tag{3}$$

Geometrically, this creates a vector field that simultaneously expands the critical edge of the target's death (to maximize d^*) and contracts the critical edges of the noise features (to minimize $d_i - b_i$). From this we devise Algorithm 1 which can be seen in appendix A.

Figures 1–4 show such optimization being applied to a noisy circle point cloud. Using the gradient descent, it optimizes (prolongs the death) an H_1 feature (a hole), while suppressing other non-target features.

Note that both λ_{target} and λ_{noise} were by default set as 1.0 in this example — these parameters can be fine-tuned to specific topologies and use-cases.

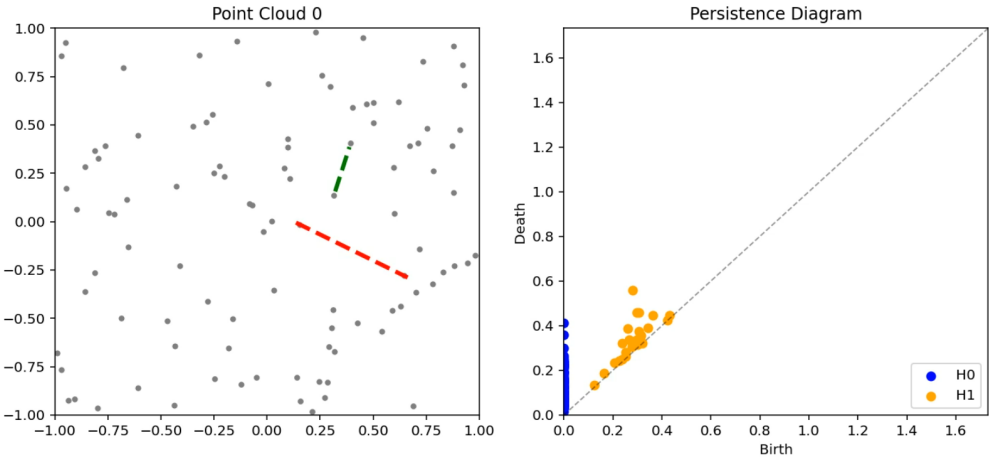

Figure 1: Iteration 0: Initial Noisy Circle.

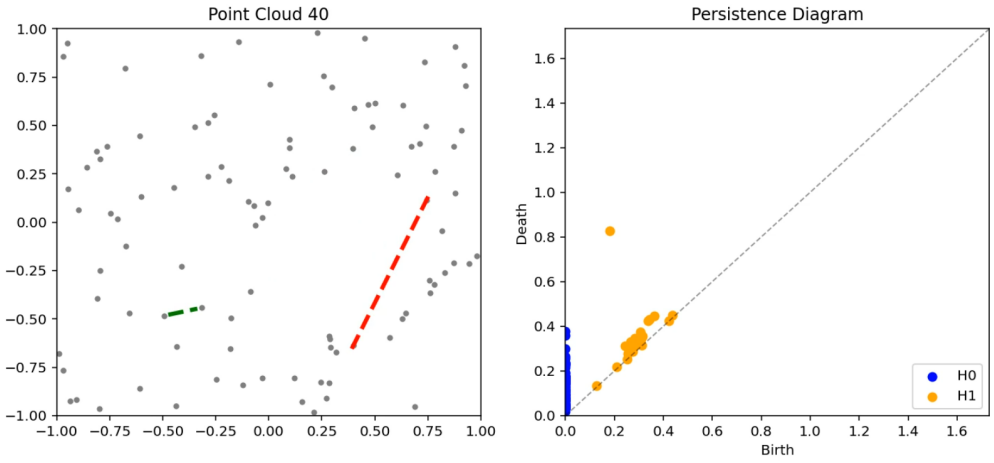

Figure 2: Iteration 40.

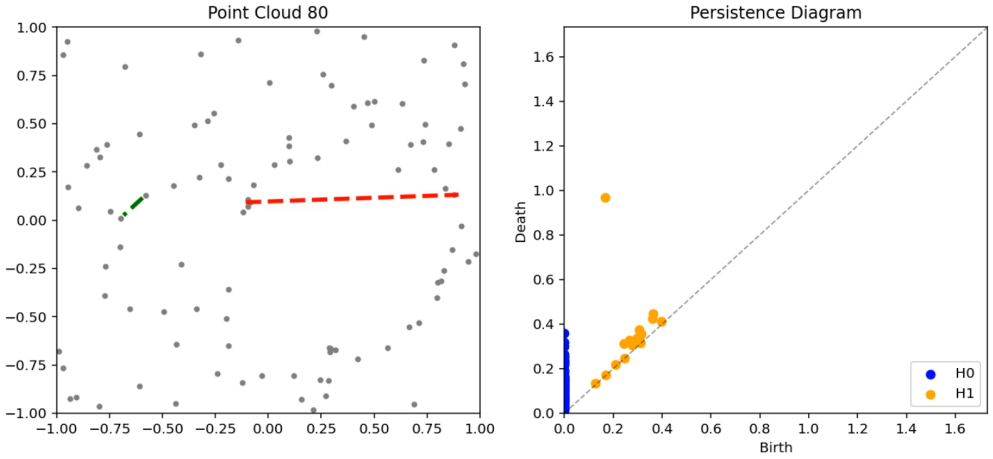

Figure 3: Iteration 80.

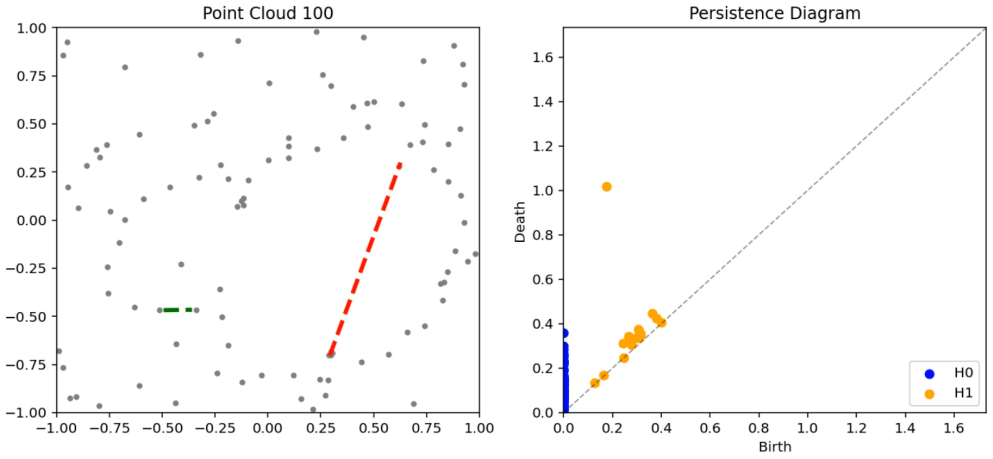

Figure 4: Iteration 100.

3 Deforming Point Clouds to a Target Signature

Our primary objective is to deform a point cloud P such that its topological signature converges toward a desired target signature. We approach this by sequentially applying the feature manipulation framework described in Section 2. The optimization objective is generalized to allow both the prolongation and shortening of topological features. Shortening is achieved by inverting the gradient sign in Eq. 3, effectively performing gradient ascent on the feature lifespan to reduce persistence.

This framework extends to 0-dimensional features (H_0) with a specific constraint. Since birth times for H_0 components are fixed at zero (representing the initial points), the optimization for connected components is restricted solely to the death critical values.

We use this method as the foundation for our deformation algorithm. To align the signatures, we sort the features of both the source and target diagrams by persistence magnitude. Features sharing the same rank index are considered "matched" pairs. The algorithm proceeds iteratively, optimizing each chosen feature toward its matched target persistence. If a matched feature does not exist in the target, the algorithm shortens the chosen feature to suppress it.

Initially, features were selected in a round-robin fashion. However, experiments showed that the point cloud often settled into a local optimum (looping among a few stable placements) without matching the target signature. To mitigate this, we adopted a stochastic strategy: selecting a random feature at each step regardless of dimension. This approach prevents the optimization from stalling in non-optimal stable states.

Figures 5, 6 and 7 the algorithm in action: the initial random configuration, the target noisy circle, and the optimized point cloud after 50,000 steps. All of them are paired with their respective topological signatures.

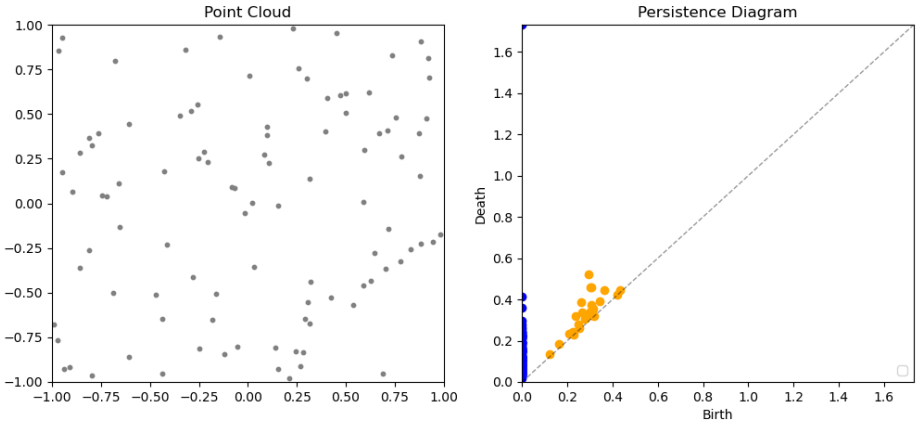

Figure 5: Initial random points.

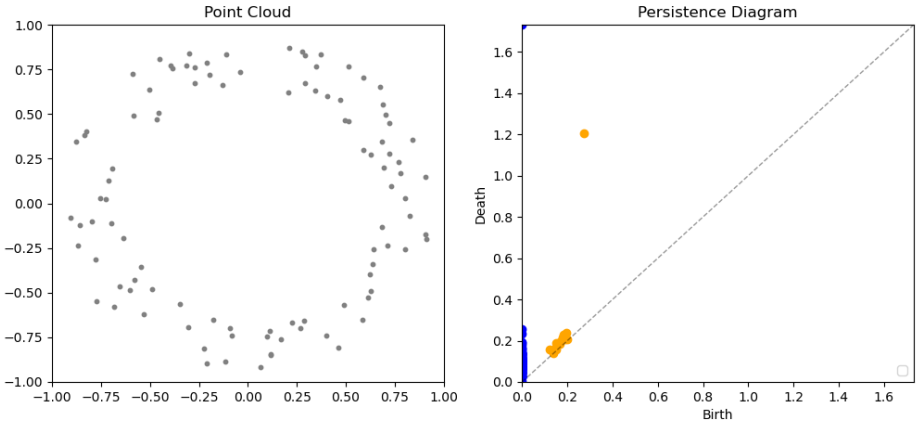

Figure 6: Target topological signature.

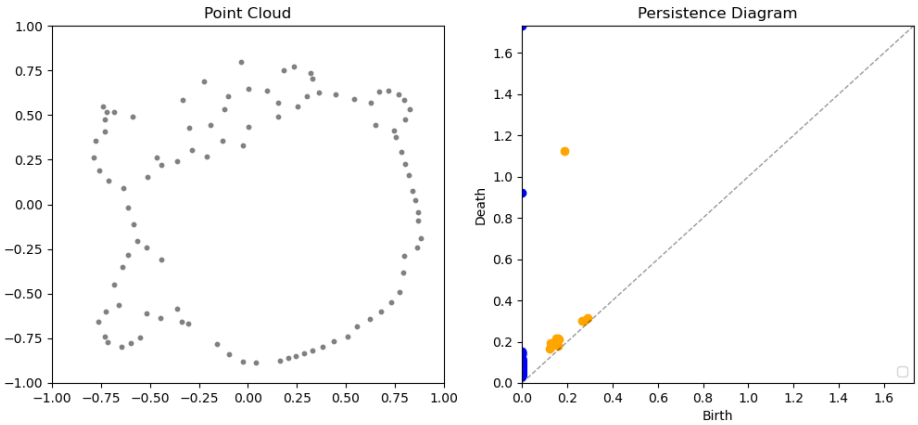

Figure 7: Optimized points after 50000 steps.

Note that the algorithm did not reach a stable position after 50000 steps, so with more steps the resulting signatures would get even closer to each other.

3.1 Comparing Results

For two-dimensional point clouds, visual inspection is often sufficient to determine similarity. However, in higher dimensions or with complex topologies, quantitative metrics are required. To compare the geometry of the point sets, we use the Hausdorff distance:

$$d_H(A, B) = \max \left\{ \sup_{a \in A} \inf_{b \in B} d(a, b), \quad \sup_{b \in B} \inf_{a \in A} d(a, b) \right\} \tag{4}$$

To compare the topological signatures (persistence diagrams), we utilize the bottleneck distance. In the experiment shown in Figures 5-7, the final Hausdorff distance is 0.908 and the bottleneck distance is 0.461.

4 Comparison with Geometric Heuristics

To validate the necessity of topological optimization, we compared our gradient-based method against a geometric heuristic, referred to here as the "Bubble Approach." This method approximates persistent features as disjoint spheres (bubbles) expanded around feature centroids, formulating the task as a dynamic sphere-packing problem.

4.1 Methodology

Let $C = \{c_1, \ldots, c_k\} \subset \mathbb{R}^2$ be the set of centroids initialized at the geometric centers of the death simplices of the k target features. The point cloud P is partitioned into clusters $P_1, \ldots, P_k$ based on the nearest centroid:

$$P_j = \{x \in P \mid j = \operatorname*{argmin}_l \|x - c_l\|\}. \tag{5}$$

Bubble Size Calculation: To ensure features remain disjoint, we define a dynamic radius r_j for each centroid c_j. The radius is constrained by the proximity to the nearest neighboring centroid and the domain boundary $\partial\Omega$:

$$r_j = \min\left(\min_{l \neq j} \frac{\|c_j - c_l\|}{2}, \quad \operatorname{dist}(c_j, \partial\Omega)\right). \tag{6}$$

Update Rules: The optimization proceeds iteratively via two coupled update steps:

1. *Point Projection:* Points $x \in P_j$ are projected onto the boundary of the bubble defined by c_j to enforce the feature shape. A smoothing term is applied to equidistant neighbors to ensure uniform distribution along the circumference:
$$x^{(t+1)} = c_j^{(t)} + r_j^{(t)} \frac{x^{(t)} - c_j^{(t)}}{\|x^{(t)} - c_j^{(t)}\|}. \tag{7}$$

2. *Centroid Drift:* Centroids move to maximize separation, governed by an inverse-square repulsive potential from other centroids and boundary constraints (treated as obstacles O_j):
$$c_j^{(t+1)} = c_j^{(t)} - \eta \sum_{o \in O_j} \frac{o - c_j^{(t)}}{\|o - c_j^{(t)}\|^2}. \tag{8}$$

4.2 Limitations

While this geometric heuristic is computationally less expensive than calculating persistent homology ($O(N)$ vs $O(N^3)$), it fails to generalize to complex topologies. Specifically, sphere packing is geometrically constrained to codimension-1 features (loops in 2D) and cannot topologically merge connected components (H_0) or selectively shorten features. These limitations underscore the necessity of the direct gradient descent approach on the persistence diagram for general feature manipulation. The following Figures 8 and 9 illustrate the working of this approach.

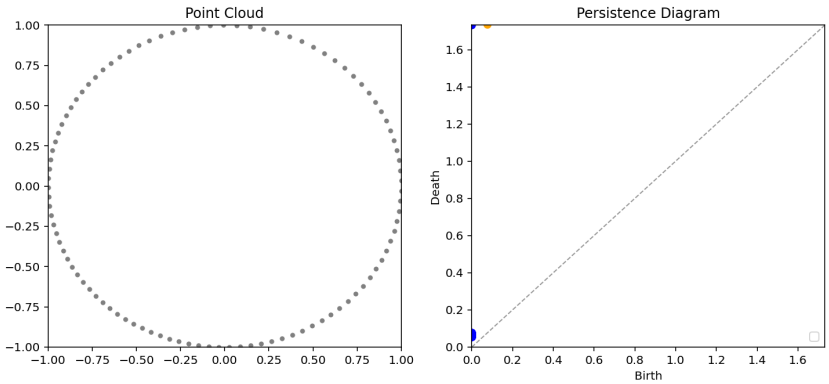

Figure 8: Optimized points to 1 centroid after 100 steps.

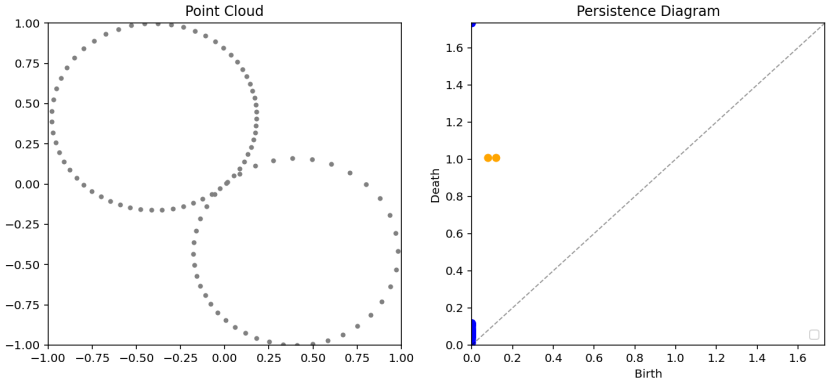

Figure 9: Optimized points to 2 centroids after 100 steps.

5 Conclusion

In this paper, we explored optimization strategies for controlling persistent homology in point clouds. We presented a theoretical framework for gradient descent on persistence diagrams and demonstrated its efficacy in manipulating single features and reconstructing target topological signatures. We contrasted this with a geometric sphere-packing heuristic, highlighting the superior flexibility of the gradient-based approach. Finally, we implemented these methods into a runnable Python program capable of generating consistent visualizations of the topological optimization process.

A Gradient Descent Algorithm and Analysis

Algorithm 1 Feature Manipulation via Persistent Homology Optimization

Input: Point cloud $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^2$, learning rate η, weights $\lambda_{\text{target}}, \lambda_{\text{noise}}$
Output: Optimized point cloud P

Build Vietoris-Rips complex $\mathcal{R}(P)$
Compute persistence for H_1 homology group
if no H_1 features exist **then**
    return P
end if
Select target feature h^* with persistence pair (b^*, d^*)
Identify noise features $\{(b_i, d_i)\}_{i \in \text{noise}}$

for each point $x_k \in P$ **do**
    Initialize $\nabla_{x_k}^{\text{total}} \leftarrow 0$
end for

Let (x_i, x_j) be the critical edge for b^*, compute $\nabla_{x_i} b^*, \nabla_{x_j} b^*$
Let (x_p, x_q) be the critical edge for d^*, compute $\nabla_{x_p} d^*, \nabla_{x_q} d^*$
$\nabla_{x_i}^{\text{total}} \leftarrow \nabla_{x_i}^{\text{total}} - \lambda_{\text{target}}(\nabla_{x_i} d^* - \nabla_{x_i} b^*)$
$\nabla_{x_j}^{\text{total}} \leftarrow \nabla_{x_j}^{\text{total}} - \lambda_{\text{target}}(\nabla_{x_j} d^* - \nabla_{x_j} b^*)$
$\nabla_{x_p}^{\text{total}} \leftarrow \nabla_{x_p}^{\text{total}} - \lambda_{\text{target}}(\nabla_{x_p} d^* - \nabla_{x_p} b^*)$
$\nabla_{x_q}^{\text{total}} \leftarrow \nabla_{x_q}^{\text{total}} - \lambda_{\text{target}}(\nabla_{x_q} d^* - \nabla_{x_q} b^*)$

for each noise feature (b_i, d_i) **do**
    Let (x_{i_1}, x_{i_2}) be the critical edge for b_i, compute $\nabla_{x_{i_1}} b_i$, $\nabla_{x_{i_2}} b_i$
    Let (x_{d_1}, x_{d_2}) be the critical edge for d_i, compute $\nabla_{x_{d_1}} d_i$, $\nabla_{x_{d_2}} d_i$
    $w_i \leftarrow 2\lambda_{\text{noise}}(d_i - b_i)$
    $\nabla_{x_{i_1}}^{\text{total}} \leftarrow \nabla_{x_{i_1}}^{\text{total}} + w_i(\nabla_{x_{i_1}} d_i - \nabla_{x_{i_1}} b_i)$
    $\nabla_{x_{i_2}}^{\text{total}} \leftarrow \nabla_{x_{i_2}}^{\text{total}} + w_i(\nabla_{x_{i_2}} d_i - \nabla_{x_{i_2}} b_i)$
    $\nabla_{x_{d_1}}^{\text{total}} \leftarrow \nabla_{x_{d_1}}^{\text{total}} + w_i(\nabla_{x_{d_1}} d_i - \nabla_{x_{d_1}} b_i)$
    $\nabla_{x_{d_2}}^{\text{total}} \leftarrow \nabla_{x_{d_2}}^{\text{total}} + w_i(\nabla_{x_{d_2}} d_i - \nabla_{x_{d_2}} b_i)$
end for

for each point $x_k \in P$ **do**
    $x_k \leftarrow x_k - \eta \nabla_{x_k}^{\text{total}}$
end for
return P

Theorem A.1 (Complexity of Algorithm 1)**.** *One iteration of Algorithm 1 runs in $O(n^9)$ time, where n is the number of points in P.*

Proof. Construction of the 2-skeleton of the Vietoris-Rips complex requires identifying all edges, of which there are $O(n^2)$, and triangles, of which there are $O(n^3)$. Let M denote the total number of simplices, thus $M \in O(n^3)$.

Computing the persistence diagram involves reducing the boundary matrix $M \times M$. The standard column algorithm takes $O(M^3)$ operations. Substituting M we get $O(O(n^3)^3) = O(n^9)$.

For the gradient calculations, the set of critical values corresponds to a subset of the edges. In the worst case, we compute gradients for $O(M)$ features. Each gradient computation takes $O(1)$ as a basic operation, hence gradient computation for each feature takes up to $O(M) = O(n^3)$. Then updating the points of P takes a linear amount of time, updating each point of the cloud; $O(n)$.

The total time is then $O(n^9) + O(n^3) + O(n)$, where the dominant term is $O(n^9)$, and is hence the upper asymptotical limit of time complexity. □

Remark. *Note that the limit in Theorem A.1 can further be improved by using a more efficient algorithms for the standard column algorithm in the boundary matrix reduction.*

Division of Work

Anja Laharnar dealt with deforming toward a target signature, its animation visualization, and the bubble algorithm. Vanja Stojanović developed single feature optimization, step by step visualization of it, and analyzed the time complexity of the algorithm.

References

[1] Mathieu Carrière et al. "Optimizing persistent homology based functions". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1294–1303.

[2] Chao Chen et al. "A topological regularizer for classifiers via persistent homology". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 2573–2582.

[3] Rickard Brüel Gabrielsson et al. "A topology layer for machine learning". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 599–609.

[4] Marcio Gameiro, Yasuaki Hiraoka, and Ippei Obayashi. "Continuation of point clouds via persistence diagrams". In: *Physica D: Nonlinear Phenomena* 334 (Nov. 2016), pp. 118–132. ISSN: 0167-2789. DOI: 10.1016/j.physd.2015.11.011. URL: http://dx.doi.org/10.1016/j.physd.2015.11.011.

[5] Yasuaki Hiraoka et al. "Topological Node2vec: Enhanced Graph Embedding via Persistent Homology". In: *arXiv preprint arXiv:2309.08241* (2023).

[6] Ippei Obayashi. "Volume-optimal cycle: Tightest representative cycle of a generator in persistent homology". In: *SIAM Journal on Applied Algebra and Geometry* 2.4 (2018), pp. 508–534.

[7] Katharine Turner et al. "Fréchet means for distributions of persistence diagrams". In: *Discrete & Computational Geometry* 52 (2014), pp. 44–70.