A Topological Regularizer for Classifiers via Persistent Homology

Chao Chen[1] **Xiuyan Ni**[2] **Qinxun Bai**[3] **Yusu Wang**[4]

[1]Stony Brook University, Stony Brook, NY [2]City University of New York, New York, NY

[3]Hikvision Research America, Santa Clara, CA [4]Ohio State University, Columbus, OH

Abstract

Regularization plays a crucial role in supervised learning. Most existing methods enforce a global regularization in a structure agnostic manner. In this paper, we initiate a new direction and propose to enforce the structural simplicity of the classification boundary by regularizing over its *topological complexity*. In particular, our measurement of topological complexity incorporates the *importance* of topological features (e.g., connected components, handles, and so on) in a meaningful manner, and provides a direct control over spurious topological structures. We incorporate the new measurement as a topological penalty in training classifiers. We also propose an efficient algorithm to compute the gradient of such penalty. Our method provides a novel way to topologically simplify the global structure of the model, without having to sacrifice too much of the flexibility of the model. We demonstrate the effectiveness of our new topological regularizer on a range of synthetic and real-world datasets.

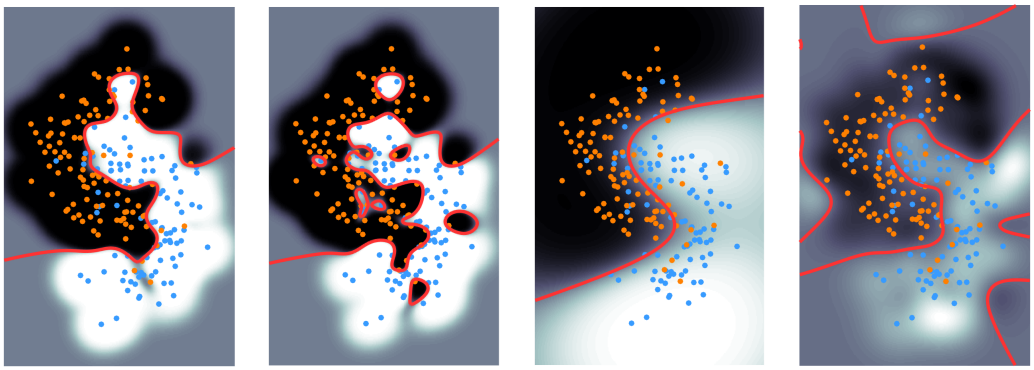
1 Introduction

Regularization plays a crucial role in supervised learning. A successfully regularized model strikes a balance between a perfect description of the training data and the ability to generalize to unseen data. A common intuition for the design of regularzers is the Occam's razor principle, where a regularizer enforces certain simplicity of the model in order to avoid overfitting. Classic regularization techniques include functional norms such as L_1 (Krishnapuram et al., 2005), L_2 (Tikhonov) (Ng, 2004) and RKHS norms (Schölkopf and Smola, 2002). Such norms produce a model with relatively less flexibility and thus is less likely to overfit.

A particularly interesting category of methods is inspired by the geometry. These methods design new penalty terms to enforce a geometric simplicity of the classifier. Some methods stipulate that similar data should have similar score according to the classifier, and enforce the smoothness of the classifier function (Belkin et al., 2006; Zhou and Schölkopf, 2005; Bai et al., 2016). Others directly pursue a simple geometry of the classifier boundary, i.e., the submanifold separating different classes (Cai and Sowmya, 2007; Varshney and Willsky, 2010; Lin et al., 2012, 2015). These geometry-based regularizers are intuitive and have been shown to be useful in many supervised and semi-supervised learning settings. However, regularizing total smoothness of the classifier (or that of the classification boundary) is not always flexible enough to balance the tug of war between overfitting and overall accuracy. The key issue is that these measurement are usually structure agnostic. For example, in Figure 1, a classifier may either overfit (as in (b)), or becomes too smooth and lose overall accuracy (as in (c)).

In this paper, we propose a new direction to regularize the "simplicity" of a classifier – Instead of using geometry such as total curvature, we directly enforce the "simplicity" of the classification boundary, by regularizing over its *topological complexity*. (Here, we take a similar functional view as Bai et al. (2016) and consider the classifier boundary as the 0-valued level set of the classifier function $f(x)$; see e.g., Figure 2.) Our measurement of topological complexity incorporates the *importance* of topological structures, e.g., connected components, handles, in a meaningful manner, and provides a direct control over spurious topological structures. This new structural simplicity can be combined with other regularizing terms (say geometry-based ones or functional norms) to train a better classifier. See Figure 1 (a) for an example, where the classifier computed with topological regularization achieves a better balance between overfitting and classification accuracy.

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.



(a) (b) (c) (d)

Figure 1: Comparison of classifiers with different regularizers on a synthetic dataset (2 moons with noise level 5%, see Sec. 4 for details). For ease of exposition, we only draw training data (blue and orange markers) and the classification boundary (red). (a): our method achieves structural simplicity without over-smoothing the classifier boundary. A standard classifier (e.g., kernel method using the same σ) could (b) overfit, or (c) overly smooth the classification boundary and reduce overall accuracy. (d): The output of the STOA method based on geometrical simplicity (Bai et al., 2016) also smooths the classifier globally.

To design a good topological regularizer, there are two key challenges. First, we want to measure and incorporate the significance of different topological structures. For example, in Figure 2 (a), we observe three connected components in the classification boundary (red). The "importance" of the two smaller components (loops) are different despite their similar geometry. The component on the left exists only due to a few training data and thus are much less robust to noise than the one on the right. Leveraging several recent developments in the field of computational topology (Edelsbrunner et al., 2000; Bendich et al., 2010, 2013), we quantify such "robustness" $\rho(c)$ of each topological structure c and define our topological penalty as the sum of the squared robustness $L_{\mathcal{T}}(f) = \sum \rho(c)^2$ over all topological structures from the classification boundary.

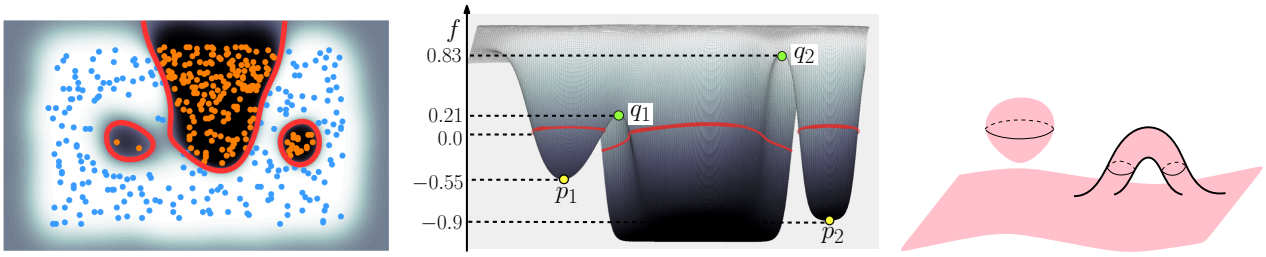
A bigger challenge is to compute the gradient of the proposed topological penalty function. In particular, the penalty function crucially depends on locations and values of *critical points* (e.g., extrema and saddles) of the classifier function. But there are no closed form solutions for these critical points. To address this issue, we propose to discretize the domain and use a piecewise linear approximation of the classifier function as a surrogate function. We prove in Section 3 that by restricting to such a surrogate function, the topological penalty is differentiable almost everywhere. We propose an efficient algorithm to compute the gradient and optimize the topological penalty. We apply the new regularizer to a kernel logistic regression model and show in Section 4 how it outperforms others on various synthetic and real-world datasets.

In summary, our contributions are as follows:

- We propose the novel view of regularizing the topological complexity of a classifier, and develop a first such topological penalty function;
- We propose a method to compute the gradient of our topological penalty. By restricting to a surrogate piecewise linear approximation of the classifier function, we prove the gradient exists almost everywhere and is tractable;
- We instantiate our topological regularizer on a kernel classifier. We provide experimental evidence of the effectiveness of the proposed method on several synthetic and real-world datasets.

For computational efficiency, in this paper, we focus on the simplest type of topological structures, i.e., connected components. The framework can be extended to more sophisticated topological structures, e.g., handles, voids, etc.

Related work. The topological summary called persistent homology Edelsbrunner et al. (2002); Carlsson and de Silva (2010) (a brief introduction of which will be provided in the supplemental material) can capture the global structural information of the data *in a multiscale manner*. It has been used in unsupervised learning, e.g., clustering (Chazal et al., 2013; Ni et al., 2017). In supervised setting, topological information has been used as powerful features. The major challenge is the metric between such topological summaries of different data is not standard Euclidean. Adams et al. (2017) proposed to directly vectorize such information. Bubenik (2015) proposed to map the topological summary into a Banach space so that statistical reasoning can be carried out (Chazal et al., 2014). To fully leverage the topological information, various kernels (Reininghaus et al., 2015; Kwitt et al., 2015; Kusano et al., 2016; Carrière et al., 2017; Zhu et al., 2016) have been proposed to approximate their distance. Hofer et al. (2017) proposed to use the topological information



(a) (b) (c)

Figure 2: (a): The classifier boundary (red curves) has two additional connected components with similar geometry. But the left one is in fact less important w.r.t. the classifier as shown in (b), where the graph of the classifier function is shown (i.e, using f as the elevation function). The left valley is easier to be removed from the 0-value level set by perturbation. (c): The classifier boundary can have different types (and dimensional) topological structures, e.g., connected components, handles, voids, etc.

as input for deep convolutional neural network. Perhaps the closest to us are (Varshney and Ramamurthy, 2015; Ramamurthy et al., 2018), which compute the topological information of the classification boundary. All these methods use topological information as an observation/feature of the data. **To the best of our knowledge, our method is the first to leverage the topological information as a prior for training the classifier.**

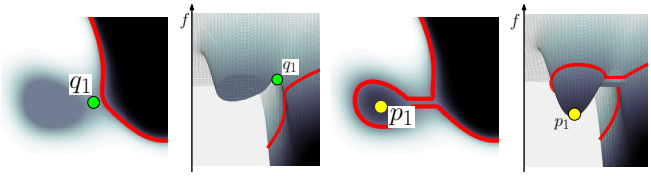
In computer vision, topological information has been incorporated as constraints in discrete optimization. Connectivity constraints can be used to improve the image segmentation quality, especially when the objects of interest are in elongated shapes. However in general, topological constraints, although intuitive, are highly complex and too expensive to be fully enforced in the optimization procedure (Vicente et al., 2008; Nowozin and Lampert, 2009). One has to resort to various approximation schemes (Zeng et al., 2008; Chen et al., 2011; Stühmer et al., 2013; Oswald et al., 2014).

2 Level Set, Topology, and Robustness

To illustrate the main ideas and concepts, we first focus on the binary classification problem[1] with a D-dimensional feature space, $\mathcal{X} \subset \mathbb{R}^D$. W.l.o.g. (without loss of generality), we assume $\mathcal{X}$ is a D-dimensional hypercube, and thus is compact and simply connected. A classifier function is a smooth scalar function, $f : \mathcal{X} \to \mathbb{R}$, and the prediction for any training/testing data $x \in \mathcal{X}$ is $\text{sign}(f(x))$. We are interested in describing the topology and geometry of the *classification boundary* of f, i.e., the boundary between the positive and negative classification regions. Formally, the boundary is the *level set* of f at value zero, i.e., the set of all points with function value zero

$$S_f = f^{-1}(0) = \{x \in \mathcal{X} | f(x) = 0\}.$$

[1]For the multilabel classification, we will use multiple one-vs-all binary classifiers (see Section 3).



(a) (b) (c) (d)

Figure 3: Two options to eliminate the left loop in Figure 2(a). Option 1: increase values of all points inside the loop so the loop disappears completely. (a): zoom-in of the new function. (b): the graph of the new function. Option 2: decrease values along a path through the saddle so the loop is merged with the U-shaped curve. (c): zoom-in of the new function, (d): the graph of the new function.

W.l.o.g., we assume S_f is a $(D-1)$-dimensional manifold, possibly with multiple connected components[2]. In Figure 2(a), the red curves represent the boundary S_f, which is a one-dimensional manifold consisting of three connected components (one U-shaped open curve and two closed loops). Note that level sets have been used extensively in the image segmentation tasks (Osher and Fedkiw, 2006; Szeliski, 2010).

For ease of exposition, we only focus on the simplest type of topological structures, i.e., connected components. For the rest of the paper, unless specifically noted, we will use "connected components" and "topological structures" interchangeably. Classification boundaries of higher dimension may have other types of topological structures, e.g., handles, voids, etc. See Figure 2(c) for the boundary of a 3D classifier. Our method can be extended to these structures.

Robustness. Our goal is to use the topological regularizer to simplify the topology of a classifier boundary. To achieve this, we need a way to rank the significance of different topological structures. The measure should be based on the underlying classifier function. To illus-

[2]The degenerate case happens if S_f passes through critical points, e.g., saddles, minima, or maxima.

trate the intuition, recall the example in Figure 2(a). To rank the three connected components of the classifier boundary S_f, simply inspecting the geometry is insufficient. The two loops have similar size. However, the left loop is less stable as it is caused by only a few training samples (two positive samples inside the loop and two negative samples between the loop and the U-shaped curve). Instead, the difference between the two loops can be observed by studying the graph of the (classifier) function (Figure 2(b), where we view the graph of the function as a terrain). Compared to the right loop, the basin inside the left loop is shallower and the valley nearby is closer to the sea level (zero), and thus it is easier to perturb the function to remove the left loop.

Intuitively, we would like to measure the significance of a component of interest, c, as the minimal amount of necessary perturbation the underlying classifier f needs in order to "shrug off" c from the zero-valued level set of the classifier function f. We measure the distance between f and its perturbed version $\hat{f}$ via the L_∞ norm, i.e., $\text{dist}(f, \hat{f}) = \max_{x \in \mathcal{X}} |f(x) - \hat{f}(x)|$.

Before we formally define our robustness measure, consider the example of Figure 2; there are two options to perturb f to remove the left loop component:

Option 1. Remove the left loop by increasing the function value of all points within the basin it encloses to $+\epsilon$, where ϵ is an infinitesimally small positive value. For the new function g, the zero-valued level set only consists of the U-shaped curve and the right loop. See Figure 3(a) and (b) for a zoomed-in view of g and its graph. In this case, the cost $\text{dist}(f, g)$ is simply ϵ plus the depth of the basin, i.e., the absolute function value of the local minimum inside the left loop (left yellow marker in Figure 2(b)), which is $\text{dist}(f, g) = |-0.55| + \epsilon = 0.55 + \epsilon$.

Option 2. Merge the left loop with the U-shaped curve by finding a path connecting them and lowering the function values along the path to $-\epsilon$. There are many paths achieving the goal. But note that all paths connecting them have to go at least as high as the nearby saddle point (left green marker in Figure 2(b)). Therefore, we choose a path passing the saddle point and has the saddle as the highest point. By changing function values along the path to $-\epsilon$, we get the new function h. In the zero-valued level set S_h of the perturbed function h, the left loop is merged with the U-shaped curve via a "pipe". See Figure 3(c) and (d) for a zoomed-in view of h and its graph. In this case, the cost $\text{dist}(f, h)$ is ϵ plus the highest height of the path, namely, the function value of the saddle point; that is, $\text{dist}(f, h) = 0.21 + \epsilon$.

To optimize the cost to remove the left loop, we choose the second option. The corresponding cost gives us the robustness of this left component c, $\rho(c) = 0.21$. Note for the right loop, its robustness is much higher as values of the associated critical points (saddle and minimum) are further away from the value zero. In fact, the minimum perturbation required is 0.83.

In this example, we observe that the robustness of a component crucially depends on the function values of two critical points, a minimum p and a saddle point q. This is not a coincidence: this pairing (p, q) is in fact a so-called *persistence pairing* computed based on the theory of persistent homology (Edelsbrunner et al., 2002; Carlsson et al., 2009). We skip the descriptions and definitions here, and refer the readers to the supplemental material for some details. Rather, we just introduce some necessary concepts:

Assume a given function $f : \mathcal{X} \to \mathbb{R}$ is a *Morse function*, i.e., a well-behaved smooth function with no degenerate critical points. Its *k-th levelset zigzag persistence diagram* dgm_k consists of a set of points $\text{dgm}_k = \{(b, d)\}$ in the plane. Intuitively, imagine we sweep the domain $\mathcal{X}$ in increasing function value $\alpha \in \mathbb{R}$, and track the change of k-dimensional topological features (as measured by the k-th homology groups) of **the levelsets** $f^{-1}(\alpha)$ through the course. Each point (b, d) indicates the *birth-time* b and *death-time* d of some topological feature in the levelset. It turns out that there exist critical points p_b and p_d of f, such that $b = f(p_b)$ and $d = f(p_d)$. Hence there is a set of pairs of critical points $\Pi_k(f) = \{(p, q)\}$, which we call *persistence pairings*, corresponding to all points in the persistence diagram dgm_Z. Each pair (p, q) indicates that a topological feature is created in the levelset when sweeping past p, and it persists till sweeping past q. The persistence of this feature is its lifetime $|f(q) - f(p)|$. Now set

$$\Pi_{S_f} := \{(p, q) \in \Pi_0(f) \mid f(p) \leq 0; f(q) \geq 0\} \quad (2.1)$$

to be the set of critical pairs corresponding to the 0-th levelset-zigzag persistence diagram created at or before function value 0, and died after it. Using results from Carlsson et al. (2009); Bendich et al. (2013), we can derive the following (see the supplemental material):

Theorem 2.1. *Let $f : \mathcal{X} \to \mathbb{R}$ be a Morse function defined on a D-dimensional hypercube $\mathcal{X} \subset \mathbb{R}^D$, and Π_{S_f} as defined above. Then there is a one-to-one correspondance π between the set of connected components of the boundary S_f and pairings in Π_{S_f}.*

Furthermore, Π_{S_f} can be computed from the so-called 0-th dimensional persistent homology induced by sublevel set filtration w.r.t. function f and w.r.t. function $-f$.

The second part of the theorem leads to an efficient algorithm to compute robustness (using existing work), which we describe shortly. From now on we identify

components in the levelset S_f with pairs of critical points in Π_{S_f}. We are now ready to define robustness.

Definition 1 (Robustness)**.** *For each connected component c from the classification boundary $S_f = f^{-1}(0)$, let $(p_c, q_c) \in \Pi_{S_f}$ be its corresponding pair of critical points. The robustness of c is $\rho(c) = \min\{|f(p_c)|, |f(q_c)|\}$.*

The robustness is closely related to but is different from the persistence. Intuitively, $f(p_c)$ indicates the time this component was first created in a levelset during the sweep of $\mathcal{X}$, and $f(q_c)$ indicates the time it is destroyed. To remove this component from the levelset $S_f = f^{-1}(0)$, we either pushes the birth-time above 0, or the death-time below 0, and the smaller cost of these two is the robustness of c. We note that this definition follows the intuition of using the well-groups to quantify robustness of levelsets (and interval levelsets) as in (Bendich et al., 2010, 2013).

In the expample in Figure 2 (b), the left loop c_1 corresponds to critical pairing (p_1, q_1) and has robustness $\rho(c_1) = |f(q_1)| = 0.21$, while the right loop corresponds to critical pairing (p_2, q_2) and has robustness $\rho(c_2) = |f(q_2)| = 0.83$.

We remark that we can define k-dimensional counterpart, $k \geq 1$, for Π_{S_f} and define the robustness for k-dimensional topological features in S_f similarly (our later topological regularization frameworks works for high-dimensional features as well). In our current algorithm, we only use the 0-th dimensional featuers as Π_{S_f} for the 0-dimensional case can be computed efficiently.

Algorithm. By Theorem 2.1, we only need to compute 0-th persistent homology induced by the sublevel-set filtration of both f and $-f$, in order to collect all necessary pairings of critical points. To do so, we need a discretization of the domain $\mathcal{X}$ and the classifier function f are evaluated at vertices of this discretization. It is known that the 0th persistent homology can be computed efficiently in time near linear to the total number of vertices and edges in the discretization (Edelsbrunner and Harer, 2010) using the union-find data structure. For completeness and to provide some intuition, we give a brief description below – we describe how to compute pairings $\widehat{\Pi}_f$ corresponding to the 0th persistence diagram induced by the sublevel-set filtration w.r.t. f. A symmetric procedure for $-f$ computes $\widehat{\Pi}_{-f}$.

For low-dimensional feature space, e.g., 2D, we take a uniform sampling of $\mathcal{X}$. A grid graph $G = (V, E)$ is built using these samples as vertices, and we can evaluate the function value of f at all vertices V. (Figure 2 in the supplementary material provides an illustration.) Next, we build a merging tree as well as a collection of pairings Π_f as follows: We sort all vertices in increasing function values $V = \{v_1, \ldots, v_n\}$. Add these vertices one-by-one in order. At any moment i, we maintain the spanning forest for all the vertices $V_i = \{v_1, \ldots, v_i\}$ that we already swept. Furthermore, each tree in the spanning forest is represented by the global minimum p_m in this tree. When two trees T_1 and T_2 (associated with global minima p_1 and p_2, respectively) merge upon the processing of node v_s, then the resulting tree is represented by the lower of the two minima p_1 and p_2, say p_1, and we add the pairing (p_2, v_2) to $\widehat{\Pi}_f$. Intuitively, the tree T_2 is created when we sweep past p_2, and is "killed" (merged to an "older" tree T_1 created at p_1 with $f(p_1) \leq f(p_2)$).

After all vertices are added, the merging tree T_f is constructed. The process can be implemented by a standard union-find data structure with a slight modification to maintain the minimum of each set in $O(m\alpha(n))$ time once the vertices are sorted (Edelsbrunner and Harer, 2010), where m is the number of edges in grid graph G, and $\alpha(n)$ is the inverse Ackermann function.

We perform the same procedure for function $-f$ to collect $\widehat{\Pi}_{-f}$. Finally, (via the proof of Theorem 2.1 in the supplemental material), the set of critical pairs w.r.t the 0-th levelset zigzag persistence diagram for f is $\Pi_0(S_f) = \widehat{\Pi}_f(S_f) \cup \widehat{\Pi}_{-f}(S_f) \cup \{(v_1, v_n)\}$, where $\widehat{\Pi}_f(S_f)$ contains all pairs (p, q) from $\widehat{\Pi}_f$ whose range covers 0 (i.e, $f(p) \leq 0 \leq f(q)$) and similarly for $\widehat{\Pi}_{-f}(S_f)$. The overall time complexity is $O(n \log n + m\alpha(n))$.

The grid-graph discretization is only feasible for low-dimensional feature space. For high dimension, we use a k-nearest-neighbor graph (KNN) $G' = (V', E)$ to represent a discretization of the domain $\mathcal{X}$: Nodes of this graph are all training data points. Thus the extracted critical points are only training data points. We then perform the same procedure to compute Π as described above using this G'.

3 Topological Penalty and Gradient

Based on the robustness measure, we will introduce our topological penalty below. To use it in the learning context, a crucial step is to derive the gradient. However, the mapping from input data to persistence pairings (Π in Theorem 2.1) is highly non-linear without an explicit analytical representation. Hence it is not clear how to compute the gradient of a topological penalty function in its original format. Our key insight is that, if we approximate the classifier function by a piecewise-linear function, then we can derive gradients for the penalty function, and perform gradient-descent optimization. Our topological penalty is implemented on a kernel logistic regression classifier, and we also show how to extend it to multilabel settings.

Given a data set $\mathcal{D} = \{(x_n, t_n) \mid n = 1, \ldots, N\}$ and

a classifier $f(x, w)$ parameterized by w, we define the objective function to optimize as the weighted sum of the per-data loss and our topological penalty.

$$L(f, \mathcal{D}) = \sum_{(x,t)\in\mathcal{D}} \ell(f(x, w), t) + \lambda L_{\mathcal{T}}(f(\cdot, w)), \quad (3.1)$$

in which λ is the weight of the topological penalty, $L_{\mathcal{T}}$. And $\ell(f(x, w), t)$ is the standard per-data loss, e.g., cross-entropy loss, quadratic loss or hinge loss.

Our topological penalty, $L_{\mathcal{T}}$, aims to eliminating the connected components of the classifier boundary. In the example of Figure 2(a), it may help eliminating both the left and the right loops, but leaving the U-shaped curve alone as it is the most robust one. Recall each topological structure of the classification boundary, c, is associated with two critical points p_c and q_c, and its robustness $\rho(c) = \min\{|f(p_c, w)|, |f(q_c, w)|\}$.

We define the **topological penalty** in Equation (3.1) as the sum of squared robustness, formally, $L_{\mathcal{T}}(f) = \sum_{c\in\mathcal{C}(S_f)} \rho(c)^2$. Here $\mathcal{C}(S_f)$ is the set of all connected components of S_f except for the most robust one. In Figure 2(a), $\mathcal{C}(S_f)$ only consists of the left and the right loops. We do not include the most robust component, as there should be at least one component left in the classifier boundary.

Gradient. A crucial yet challenging task is to compute the gradient of such topological penalty. In fact, there has not been any gradient computation for topology-inspired measurement. A major challenge is the lack of a closed form solution for the critical points of any non-trivial function. Previous results show that even a simple mixture of isotropic Gaussians can have exponentially many critical points (Edelsbrunner et al., 2013; Carreira-Perpiñán and Williams, 2003).

In this paper, we propose a solution that circumvents the direct computation of critical points in the continuous domain. The key idea is to use a piecewise linear approximation of the classifier function. Recall we discretize the feature space into a grid or kNN graph, $G = (V, E)$, and only evaluate classifier function values at a finite set of locations/points. Now consider the piecewise linear function $\hat{f}$ which agrees with f at all sample points in V, but linearly interpolates along edges. We show that restricting to such piecewise linear functions, the gradient of $L_{\mathcal{T}}$ is indeed computable.

Theorem 3.1. *Using the piecewise linear approximation $\hat{f}$, the topological penalty $L_{\mathcal{T}}(\hat{f}(\cdot, w))$ is differentiable almost everywhere over the space of piecewise linear functions.*

Proof. For the piecewise linear approximate $\hat{f}$, all critical points have to come from the set of vertices V of the discretization. Their pairing and correspondence to the connected components can be directly computed using the algorithm in Section 2.

We first assume $\hat{f}$ has unique non-zero function values at all points in V, i.e, $\hat{f}(u) \neq \hat{f}(v)$, $\forall u, v \in V$. Let Δ be the lowerbound of the difference between the absolute function values of elements in V, as well as the absolute function values of all vertices: $\Delta = \min\{\min_{u,v\in V, u\neq v} ||\hat{f}(u)| - |\hat{f}(v)||, \min_{u\in V} |\hat{f}(u)|\}$ To prove our theorem, we show that there exists a small neighborhood of the function $\hat{f}$, so that for any function in this neighborhood, the critical points and their pairings remain unchanged. To see this, we note that the any function in such neighborhood of $\hat{f}$ is also piecewise linear functions realized on the same graph G. We define the neighborhood to be a radius $\Delta/2$ open ball in terms of L_∞ norm, formally, $\mathcal{F} = \{g \mid |\hat{f}(v) - g(v)| < \Delta/2, \forall v \in V\}$. For any $g \in \mathcal{F}$, $\forall u, v \in V$, we have the following three facts:

Fact 1. $\hat{f}(u) < \hat{f}(v)$ if and only if $g(u) < g(v)$.

Fact 2. $\hat{f}(u) < 0$ if and only if $g(u) < 0$, and $\hat{f}(u) > 0$ if and only if $g(u) > 0$

Fact 3. $|\hat{f}(u)| < |\hat{f}(v)|$ if and only if $|g(u)| < |g(v)|$.

The first two facts imply that the ordering of elements in V induced by their function values are the same for g and $\hat{f}$. Note the second condition is necessary as it guarantees the optimal option to remove a component are the same for $\hat{f}$ and g. Consequently, the filtration of all elements of G induced by g and $\hat{f}$ are the same. By definition of persistent homology, the persistence pairs (of critical points) are identical for g and $\hat{f}$. In other words, the pair associated with each connected component c, (p_c, q_c) are the same for both g and $\hat{f}$.

Furthermore, the third condition guarantees that for each c, $g(p_c) < g(q_c)$ if and only if $\hat{f}(p_c) < \hat{f}(q_c)$. If for $\hat{f}$, p_c is the critical point that accounts for the robustness, i.e., $\rho(c) = \hat{f}(p_c)$, then p_c also accounts for $\rho(c)$ for function g. Denote by p_c^* as such critical point. $p_c^* = \text{argmin}_{p\in\{p_c,q_c\}}\{|\hat{f}(p_c)|, |\hat{f}(q_c)|\} = \text{argmin}_{p\in\{p_c,q_c\}}\{|g(p_c)|, |g(q_c)|\}$. Thus $\rho(c) = |\hat{f}(p_c^*)|$, in which the critical point p_c^* remains a constant for any g within the small neighborhood of $\hat{f}$.

With constant p_c^*'s, and knowing that $\hat{f}$ and f agree at all elements of V, the gradient is straightforward

$$\nabla_w L_{\mathcal{T}} = \sum_{c\in\mathcal{C}(S_f)} \nabla_w(\rho(c)^2) = \sum_{c\in\mathcal{C}(S_f)} 2f(p_c^*, w)\frac{\partial f(p_c^*)}{\partial w}.$$

Note that this gradient is intractable without the surrogate piecewise linear function $\hat{f}$; for the original classifier f, p_c^* changes according to w in a complex manner.

Finally, note that elements in V may have the save function values or 0 values. In such cases, p_c^* may

not be uniquely defined and the gradient does not exist. However, these events constitute a measure zero subspace of functions and do not happen generically. In other words, $L_{\mathcal{T}}(\hat{f})$ is a piecewise smooth loss function over the space of all piecewise linear functions. It is differentiable *almost everywhere*. □

Intuition of the gradient. During the optimization process, we take the opposite direction of the gradient, i.e., $-\nabla_w L_{\mathcal{T}}$. For each component $c \in \mathcal{C}(S_f)$, taking the direction $-\nabla_w(\rho(c)^2)$ is essentially pushing the function value of the critical point p_c^* closer to zero. In the example of Figure 2, for the left loop, the gradient decent will push the function value of the saddle point (left green marker) closer to zero, effectively dragging the path down as in Figure 3(c) and (d). If it is the case when p_c^* is the minimum, the gradient descent will increase the function value of the minimum, effectively filling the basin as in Figure 3(a) and (b).

Instantiating on kernel machines. In principle, our topological penalty can be incorporated with any classifier. Here, we combine it with a kernel logistic regression classifier to demonstrate its advantage. We first present details for a binary classifier. We will extend it to multilabel settings. For convenience, we abuse the notation, drop $\hat{f}$ and only use f.

In a kernel logistic regression, the prediction function is $f(x,w) = g\left(\phi(x)^T w\right) = 1/\left(1+\exp\left(-\phi(x)^T w\right)\right)$. The N-dim feature $\phi(x) = (k(x,x_1),\cdots,k(x,x_N))^T$ consists of the Gaussian kernel distance between x and the N training data. The per-data loss $\ell(f(x,w),t)$ is the standard cross-entropy loss and its gradient can be found in a standard textbook (Bishop, 2006).

Next we derive the gradient for the topological penalty. First we need to modify the classifier slightly. Notice the range of f is between zero and one, and the prediction is $\text{sign}(f-0.5)$. To fit our setting in which the zero-valued level set is the classification boundary, we use a new function $\tilde{f} = f - 0.5$ as the input for the topological penalty. The gradient is

$$\nabla_w L_{\mathcal{T}} = \sum_{c\in\mathcal{C}(S_f)} 2\tilde{f}(p_c^*,w)\frac{\partial \tilde{f}(p_c^*)}{\partial w}$$
$$= \sum_{c\in\mathcal{C}(S_f)} \left(-2f(p_c^*,w)^3 + 3f(p_c^*,w)^2 - f(p_c^*,w)\right)\phi(p_c^*)$$

Our overall algorithm repeatedly computes the gradient of the objective function (gradient of cross-entropy loss and gradient of topological penalty), and update the parameters w accordingly, until it converges. At each iteration, to compute the gradient of the topological penalty, we compute the critical point p_c^*'s for all connected components via the algorithm of Section 2.

Multilabel settings. For multilabel classification with K classes, we use the multinomial logistic regression classifier $f^k(x,W), k = 1..K$ with parameters $W = (w^1,\cdot,w^K)$. The per-data loss is again the standard cross-entropy loss. For the topological penalty, we create K different scalar functions $\psi^k(x,W) = \max_{t\neq k} f^t(x,W) - f^k(x,W)$. If $\psi^k(x,W) < 0$, we classify x as label k. The 0-valued level set of $\psi^k(x,W)$ is the classification boundary between label k and all others. Summing the total robustness over all different ψ^k's give us the multilabel topological penalty. We omit the derivation of the gradients due to space constraint. The computation is similar to binary-labeled setting, except that at each iteration, we need to compute the persistence pairs for all the K functions.

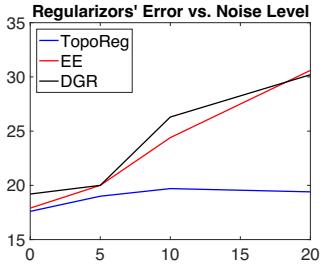


Figure 4: Comparison of different regularizers with different noise level. TopoReg is more robust to the noise compared with other geometric regularizers.

4 Experiments

We test our method (TopoReg) on multiple synthetic datasets and real world datasets. The weight of the topological penalty, λ and the Gaussian kernel width σ are turned via cross-validation. To compute topological information requires discretization of the domain. For 2D data, we normalize the data to fit the unit square $[0,1]\times[0,1]$, and discretize the square into a grid with $300\times$ 300 vertices. For high-dimensional data, we use the KNN graph with $k = 3$.

Baselines. We compare our method with several baselines: k-nearest-neighbor classifier (KNN), logistic regression (LG), Support Vector Machine (SVM), and Kernel Logistic Regression (KLR) with functional norms (L_1 and L_2) as regularizers. We also compare with two state-of-the-art methods based on geometric-regularizers: the Euler's Elastica classifier (EE) (Lin et al., 2015) and the Classifier with Differential Geometric Regularization (DGR) (Bai et al., 2016). All relevant hyperparameters are tuned using cross-validation.

For every dataset and each method, we randomly divide the datasets into 6 folds. Then we use each of the 6 folds as testing set, while doing a 5-fold cross validation on the rest 5 folds data to find the best hyperparameters. Once the best hyperparameters are found, we train on the entire 5 folds data and test on the testing set.

Table 1: The mean error rate of different methods.

Synthetic							
	KNN	LG	SVM	EE	DGR	KLR	TopoReg
Blob-2 (500,5)	7.61	8.20	7.61	8.41	7.41	7.80	**7.20**
Moons (500,2)	20.62	20.00	19.80	19.00	19.01	18.83	**18.63**
Moons (1000,2,Noise 0%)	19.30	19.59	19.89	17.90	19.20	17.80	**17.60**
Moons (1000,2,Noise 5%)	21.60	19.29	19.59	22.00	22.30	**19.00**	**19.00**
Moons (1000,2,Noise 10%)	21.10	**19.19**	19.89	24.40	26.30	20.00	19.70
Moons (1000,2,Noise 20%)	23.00	19.79	**19.40**	30.60	30.20	19.50	**19.40**
AVERAGE	18.87	17.68	17.70	20.39	20.74	21.63	**16.92**
UCI							
	KNN	LG	SVM	EE	DGR	KLR	TopoReg
SPECT (267,22)	17.57	17.20	18.68	**16.38**	23.92	18.31	17.54
Congress (435,16)	5.04	4.13	4.59	4.59	4.80	**4.12**	4.58
Molec. (106,57)	24.54	19.10	19.79	17.25	16.32	19.10	**12.62**
Cancer (286,9)	29.36	28.65	28.64	28.68	31.42	29.00	**28.31**
Vertebral (310,6)	15.47	15.46	23.23	17.15	13.56	12.56	**12.24**
Energy (768,8)	0.78	0.65	0.65	0.91	0.78	**0.52**	**0.52**
AVERAGE	15.46	14.20	15.93	14.16	15.13	13.94	**11.80**
Biomedicine							
	KNN	LG	SVM	EE	DGR	KLR	TopoReg
KIRC (243,166)	30.12	28.87	32.56	31.38	35.50	31.38	**26.81**
fMRI (1092,19)	46.70	74.91	74.08	82.51	**31.32**	34.07	33.24

Data. In order to thoroughly evaluate the behavior of our model, especially in large noise regime, we created synthetic data with various noise levels. Beside feature space noise, we also inject different levels of label noise, e.g., randomly perturb labels of 0%, 5%, 10% and 20% of the training data. We also evaluate our method on real world data. We use several UCI datasets with various sizes and dimensions to test our method (Lichman, 2013). In addition, we use two biomedical datasets. The first is the kidney renal clear cell carcinoma cancer (KIRC) dataset (Yuan et al., 2014) extracted from the Cancer Genome Atlas project (TCGA) (Sharpless and others, 2018). The features of the dataset are the protein expression measured on the MD Anderson Reverse Phase Protein Array Core platform (RPPA). The second dataset is a task-evoked functional MRI images, which has 19 dimensions (corresponding to activities at 19 brain ROIs) and 6 labels (corresponding to 6 different tasks) (Ni et al., 2018).

The results are reported in Table 1. We also report the average performance over each category (AVERAGE). The two numbers next to each dataset name are the data size N and the dimension D, respectively. The average running time over all the datasets for our method is 2.08 seconds.

Discussions. Our method generally outperforms existing methods on datasets in Table 1. More importantly, we note that our method also provides best or close to best performance among all approaches tested. (For example, while EE performs well on some datasets, its performance can be significantly worse than the best for some other datasets.)

On synthetic data, we found that TopoReg has a bigger advantage on relatively noisy data. This is expected. Our method provides a novel way to topologically simplify the global structure of the model, without having to sacrifice too much of the flexibility of the model. Meanwhile, to cope with large noise, other baseline methods have to enforce an overly strong global regularization in a structure agnostic manner. We also observe that TopoReg performs relatively stable when label noise is large, while the other geometric regularizers are much more sensitive to label noise. See Figure 4 for a comparison. We suspect this is because the other geometric regularizers are more sensitive to the initialization and tend to stuck in bad local optima.

Finally, the idea of topological regularizor is general and can be potentially applied to unsupervised context. We also believe that the topological penalty is in general not convex. We note in a parallel work, Poulenard et al. (2018) proposed to optimize persistent-homology-inspired objective function for shape matching.

Acknowledgements

We thank reviewers for their insightful comments and suggestions. This work was partially supported by NSF IIS-1855759, IIS-1815697, CCF-1855760, CCF-1733798, and CCF-1740761.

References

Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., and Ziegelmeier, L. (2017). Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252.

Bai, Q., Rosenberg, S., Wu, Z., and Sclaroff, S. (2016). Differential geometric regularization for supervised learning of classifiers. In *International Conference on Machine Learning*, pages 1879–1888.

Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434.

Bendich, P., Edelsbrunner, H., and Kerber, M. (2010). Computing robustness and persistence for images. *IEEE transactions on visualization and computer graphics*, 16(6):1251–1260.

Bendich, P., Edelsbrunner, H., Morozov, D., Patel, A., et al. (2013). Homology and robustness of level and interlevel sets. *Homology, Homotopy and Applications*, 15(1):51–72.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. springer New York.

Bubenik, P. (2015). Statistical topological data analysis using persistence landscapes. *The Journal of Machine Learning Research*, 16(1):77–102.

Cai, X. and Sowmya, A. (2007). Level learning set: A novel classifier based on active contour models. In *Proc. European Conf. on Machine Learning (ECML)*, pages 79–90.

Carlsson, G. and de Silva, V. (2010). Zigzag persistence. *Foundations of Computational Mathematics*, 10(4):367–405.

Carlsson, G., de Silva, V., and Morozov, D. (2009). Zigzag persistent homology and real-valued functions. In *Proc. 25th Annu. ACM Sympos. Comput. Geom.*, pages 247–256.

Carreira-Perpiñán, M. Á. and Williams, C. K. (2003). On the number of modes of a gaussian mixture. In *International Conference on Scale-Space Theories in Computer Vision*, pages 625–640. Springer.

Carrière, M., Cuturi, M., and Oudot, S. (2017). Sliced wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning*, pages 664–673.

Chazal, F., Glisse, M., Labruère, C., and Michel, B. (2014). Convergence rates for persistence diagram estimation in topological data analysis. In *International Conference on Machine Learning (ICML)*, pages 163–171.

Chazal, F., Guibas, L. J., Oudot, S. Y., and Skraba, P. (2013). Persistence-based clustering in riemannian manifolds. *Journal of the ACM (JACM)*, 60(6):41.

Chen, C., Freedman, D., and Lampert, C. H. (2011). Enforcing topological constraints in random field image segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2089–2096. IEEE.

Edelsbrunner, H., Fasy, B. T., and Rote, G. (2013). Add isotropic gaussian kernels at own risk: More and more resilient modes in higher dimensions. *Discrete & Computational Geometry*, 49(4):797–822.

Edelsbrunner, H. and Harer, J. (2010). *Computational Topology: an Introduction*. AMS.

Edelsbrunner, H., Letscher, D., and Zomorodian, A. (2000). Topological persistence and simplification. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 454–463. IEEE.

Edelsbrunner, H., Letscher, D., and Zomorodian, A. (2002). Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533.

Hofer, C., Kwitt, R., Niethammer, M., and Uhl, A. (2017). Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, pages 1633–1643.

Krishnapuram, B., Carin, L., Figueiredo, M., and Hartemink, A. (2005). Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE. Trans. Pattern. Anal. Mach. Intell*, 32.

Kusano, G., Hiraoka, Y., and Fukumizu, K. (2016). Persistence weighted gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, pages 2004–2013.

Kwitt, R., Huber, S., Niethammer, M., Lin, W., and Bauer, U. (2015). Statistical topological data analysis-a kernel perspective. In *Advances in neural information processing systems*, pages 3070–3078.

Lichman, M. (2013). UCI machine learning repository.

Lin, T., Xue, H., Wang, L., Huang, B., and Zha, H. (2015). Supervised learning via euler's elastica models. *Journal of Machine Learning Research*, 16:3637–3686.

Lin, T., Xue, H., Wang, L., and Zha, H. (2012). Total variation and Euler's elastica for supervised learning. *Proc. International Conf. on Machine Learning (ICML)*.

Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

Ni, X., Quadrianto, N., Wang, Y., and Chen, C. (2017). Composing tree graphical models with persistent homology features for clustering mixed-type data. In *International Conference on Machine Learning*, pages 2622–2631.

Ni, X., Yan, Z., Wu, T., Fan, J., and Chen, C. (2018). A region-of-interest-reweight 3d convolutional neural network for the analytics of brain information processing. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 302–310. Springer.

Nowozin, S. and Lampert, C. H. (2009). Global connectivity potentials for random field models. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 818–825. IEEE.

Osher, S. and Fedkiw, R. (2006). *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media.

Oswald, M. R., Stühmer, J., and Cremers, D. (2014). Generalized connectivity constraints for spatio-temporal 3d reconstruction. In *European Conference on Computer Vision*, pages 32–46. Springer.

Poulenard, A., Skraba, P., and Ovsjanikov, M. (2018). Topological function optimization for continuous shape matching. In *Computer Graphics Forum*, volume 37, pages 13–25. Wiley Online Library.

Ramamurthy, K., Varshney, K. R., and Mody, K. (2018). Topological data analysis of decision boundaries with applicaiton to model selection.

Reininghaus, J., Huber, S., Bauer, U., and Kwitt, R. (2015). A stable multi-scale kernel for topological machine learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4741–4748.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Sharpless, N. E. and others (2018). TCGA: The Cancer Genome Atlas. Accessed: 10/01/2018.

Stühmer, J., Schröder, P., and Cremers, D. (2013). Tree shape priors with connectivity constraints using convex relaxation on general graphs. In *ICCV*, volume 13, pages 1–8.

Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

Varshney, K. and Willsky, A. (2010). Classification using geometric level sets. *Journal of Machine Learning Research*, 11:491–516.

Varshney, K. R. and Ramamurthy, K. N. (2015). Persistent topology of decision boundaries. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 3931–3935. IEEE.

Vicente, S., Kolmogorov, V., and Rother, C. (2008). Graph cut based image segmentation with connectivity priors. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE conference on*, pages 1–8. IEEE.

Yuan, Y., Van Allen, E. M., Omberg, L., Wagle, N., Amin-Mansour, A., Sokolov, A., Byers, L. A., Xu, Y., Hess, K. R., Diao, L., et al. (2014). Assessing the clinical utility of cancer genomic and proteomic data across tumor types. *Nature biotechnology*, 32(7):644.

Zeng, Y., Samaras, D., Chen, W., and Peng, Q. (2008). Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in n–d images. *Computer vision and image understanding*, 112(1):81–90.

Zhou, D. and Schölkopf, B. (2005). Regularization on discrete spaces. In *Pattern Recognition*, pages 361–368. Springer.

Zhu, X., Vartanian, A., Bansal, M., Nguyen, D., and Brandl, L. (2016). Stochastic multiresolution persistent homology kernel. In *IJCAI*, pages 2449–2457.