Topological optimization of the decision boundary

Noam Ghenassia

November 30, 2023

Abstract

In this work, we introduce `TopOpt`, an algorithm that allows enforcing topological restrictions on the decision boundary of a neural network by optimizing functions of the persistence diagram of a simplicial complex constructed on a sampling of it. We also propose the normal metric filtration, a modified version of the Vietoris-Rips filtration that allows to better capture the topology of an embedded manifold when the normal vectors of the manifold are known.

`TopOpt` can keep track of the gradient of the topological loss with respect to the parameters of the neural network, thanks to the implicit differentiation tool provided in Blondel et al. 2021. Therefore, `TopOpt` is able to optimize the parameters of the neural network such that the decision boundary satisfies certain topological properties.

We illustrate how `TopOpt` works on various examples, including a one-parameter toy model and a dense neural network. Our results show that `TopOpt` is able to successfully optimize the decision boundary of a neural network while enforcing topological invariants. Furthermore, we show how to use `TopOpt` to enforce topological properties of 3d objects that are modelled as levelsets of a neural network.

Introduction

Deep learning classification models are trained by optimizing a loss function that describes the failure of the model to correctly classify samples from a training set. This training set is drawn from an underlying unknown distribution, and the aim of the neural network is to fit this distribution. Therefore, a good model should be able to correctly classify unseen examples drawn from the same distribution as the one it was trained on.

This goal motivated recent work, that aims to not only minimize the training error, but also the generalization error, by minimizing quantities related to it, such as sharpness (Foret et al. 2020) and adaptive sharpness (Kwon et al. 2021). However, these approaches suffer from several issues. First, they do not provide clear insights about the induced modifications of the behaviour of the network in the latent space, therefore leading to poor interpretability. Second, criteria such as sharpness do not allow to distinguish between the regions of the latent space that are robustly classified by the network from regions that suffer from overfitting. This observation begs the question: how can we optimize the specific weights of the neural network that are responsible for overfitting? More precisely, can we modify the weights in order to selectively modify the decision boundary in regions that fitted the noise rather than the actual distribution?

Here, we propose `TopOpt` [1], an algorithm that performs this task by tracking the topology of the decision boundary. Specifically, it samples the decision boundary and constructs a filtered simplicial complex on it. It then computes the persistent homology of the complex, which then allows to compute functions of the persistent homology, according to the computation proposed in Carrière et al. 2020. Crucially, the algorithm is able to keep track of the gradient of the topological loss with respect to the weights of the neural network, thanks to the implicit differentiation tool provided in Blondel et al. 2021. Therefore, `TopOpt` is able to optimize specifically the weights of the neural

[1] the code is available at https://github.com/Noam-Ghenassia/dbopt

network that are responsible for specific topological features of the decision boundary, such as additional cycles that might correspond to over-fitting.

Our contributions are the following :

- We introduce `TopOpt`, an algorithm for optimizing functions of persistent homology of the decision boundary of a neural network.
- We propose a modification of the metric of the simplicial complex used to compute the Vietoris Rips filtration, that leverages the normal vectors of the decision boundary, which represents additional information compared to the usual setting of persistent homology, and is computable by taking the gradient of the difference of the logits of the network.
- We illustrate how `TopOpt` works on some toy models and on a small dense neural network. We then show how it allows to impose topological restrictions on implicit representations of 3D objects as level sets.

1 Background

1.1 Decision boundary

In the setting of binary classification, neural networks are parameterized functions $f_w : \mathbb{R}^d \to \mathbb{R}^2 : x \mapsto \hat{y}$, where d is the input dimension (i.e., the number of features of the data-points), and w are the parameters (weights and biases) of the network. The output $\hat{y} = (\hat{y_1}, \hat{y_2})$ of the network is the vector of the logits, which can be transformed into probabilities for each class using the softmax function. A given point in the input space is classified based on the logits, that is, the predicted class is the one with the highest logit. Therefore, we define the *decision boundary* as follows:

$$\mathcal{M} = \{x \in \mathbb{R}^d : \hat{y} = f(x), \hat{y_1} = \hat{y_2}\}$$

It partitions the input space in regions with the same predicted class.

1.2 Persistence diagrams

Persistent homology is a central tool in topological data analysis. It allows to approximate the homology of a manifold from a point cloud sampled from it. It proceeds by constructing a filtered simplicial complex on the point cloud, and computing its simplicial homology across filtration values. Features that span large intervals of the filtration are considered more significant.

Definition 1.1 (Simplex, face and simplicial complex)**.** The $n-1$ dimensional simplex

$$\sigma = \left\{ \sum_{i=1}^{n} a_i x_i \mid a_i > 0 \; \forall \; i \in \{1, ..., n\}, \sum_{i=1}^{n} a_i = 1 \right\}$$

is the convex hull of the vertices $\mathcal{V}_\sigma = \{x_i \in \mathbb{R}^d\}_{i \in \{1,...,n\}}$. A face of σ is any simplex σ' with vertices $V_{\sigma'} \subset \mathcal{V}_\sigma$. A simplicial complex $\mathcal{K}$ is a set of simplices such that for each $\sigma \in \mathcal{K}$, each face of σ is a simplex of $\mathcal{K}$, and for each $\sigma_1, \sigma_2 \in \mathcal{K}$, $\sigma_1 \cap \sigma_2$ is a face of σ_1 and σ_2.

The aim of persistent homology is to retrieve the topology of a simplicial complex that resembles the underlying manifold $\mathcal{M}$. To do that, we add simplices with vertices in the point cloud sampled from $\mathcal{M}$. However, not all subsets of the point cloud span relevant simplices of the complex. Therefore, we need a procedure to determine which simplices are involved in relevant topological features.

Definition 1.2 (Filtration)**.** Let $\mathcal{E}$ be an ordered set. Then $\{\mathcal{K}_\epsilon\}_{\epsilon \in \mathcal{E}}$ is a filtration of the simplicial complex $\mathcal{K}$ if $\mathcal{K}_{\epsilon_i} \subseteq \mathcal{K}_{\epsilon_j}$ for all $\epsilon_i \leq \epsilon_j$ and $\bigcup_{\epsilon \in \mathcal{E}} \mathcal{K}_\epsilon = \mathcal{K}$. Equivalently, a filtration of $\mathcal{K}$ can be seen as a vector $\Phi = (\Phi_\sigma)_{\sigma \in \mathcal{K}}$ where each entry Φ_σ is the filtration value of σ, that is, $\Phi_\sigma = \min_{r \in \{1,...,m\}}\{r \mid \sigma \in \mathcal{K}_{\epsilon_r}\}$.

Definition 1.3 (Vietoris-Rips complex)**.** Let $\mathcal{P}$ be a subset of points in $\mathbb{R}^d$ and ϵ a positive real value. The simplicial complex $\mathcal{K}_\epsilon$ is the minimal simplicial complex such that for each simplex $\sigma \in \mathcal{K}_\epsilon$, $||x_i - x_j|| < 2\epsilon$ for all $x_i, x_j \in \sigma$.

To retrieve the topology of a manifold $\mathcal{M}$ given a set of points sampled from it, we construct a filtered Vietoris-Rips complex on the point cloud, and record the homology of the complex at each filtration level.

Definition 1.4 (Vector space of cycles, boundary map and Simplicial homology)**.** Let $\mathcal{K}$ be a simplicial complex and $\mathbb{F}$ be a field. If $\mathcal{K}_k$ is the set of k dimensional simplices in $\mathcal{K}$, then we can define the vector space of k dimensional cycles

$$V_k = \left\{ \sum a_i x_i \mid a_i \in \mathbb{F}, x_i \in \mathcal{K}_k \right\}$$

Let σ be a simplex with vertices $\mathcal{V}_\sigma = (v_1, v_2, ..., v_k)$. The boundary map is the group homomorphism defined by

$$\partial_k \colon V_k \to V_{k-1} \colon \partial_k(\sigma) = \sum_{i=0}^{k} (-1)^i (v_0, ..., \hat{v_i}, ..., v_k)$$

where $(v_0, ..., \hat{v_i}, ..., v_k)$ is the face spanned by $\mathcal{V}_\sigma \setminus \{v_i\}$. It sends a simplex on a linear combinations of its codimension 1 faces. The k-dimensional homology group of a simplicial complex $\mathcal{K}$ is

$$H_k(\mathcal{K}) = \operatorname{Ker} \partial_k(\mathcal{K}) / \operatorname{Im} \partial_{k+1}(\mathcal{K})$$

By construction, $\operatorname{Im} \partial_{k+1} \subseteq \operatorname{Ker} \partial_k(\mathcal{K})$. The homology in dimension k captures the failure of this inclusion to be an equality: $\operatorname{Ker} \partial_k(\mathcal{K})$ represents the k-dimensional cycles formed by k-dimensional simplices, and $\operatorname{Im} \partial_{k+1}$ are those of these cycles that are part of the boundary of $k+1$-dimensional simplex. Hence, the quotient contains cycles up to boundaries. In other words, H_k is a vector space with generators the simplices that form the boundary of a k-dimensional hole in $\mathcal{K}$.

To retrieve topological information at all scales, we construct a Vietoris-Rips filtration of the point cloud sampled from the manifold, and record the homology at each filtration value ϵ. The information extracted can be summarized in a persistence diagram.

Definition 1.5 (Persistent pair and Persistence diagram)**.** A k-dimensional persistent pair is a pair $(\sigma_{l(j)}, \sigma_j)$ such that $\sigma_{l(j)}$ creates a feature (i.e., that generates a k-dimensional cycle) in k-dimensional homology, and σ_j destroys it (i.e., that fills the cycle). Some of the features may not be destroyed during the filtration, this yields persistent pairs (σ, ∞).
The k-dimensional persistence diagram of the filtered complex $\mathcal{K}$ is given by

$$D_k = \bigcup_{i=1}^{p} (\Phi_{\sigma_{l(i)}}, \Phi_{\sigma_i}) \cup \bigcup_{j=1}^{q} (\Phi_{\sigma_j}, \infty)$$

where p is the number of features with a finite lifetime, and q is the number of features that don't disapear during the filtration. The diagram is obtained by associating each k-dimensional persistent pair to its filtration values. The persistence diagram D is the disjoint union of the k-dimensional persistence diagrams.

As explained in Carrière et al. 2020, an important consequence of definition 1.5 is that persistence diagrams are only permutations of the simplicies. Indeed, a filtration induces a total preorder on the simplices: $\sigma \preceq \tau$ if $\Phi_\sigma \leq \Phi_\tau$. Simplices with the same filtration value can be ordered according to some arbitrary rule to get a total order on the simplices that is compatible with the filtration. The persistence diagram can therefore be seen as a reordering of the simplices that arranges them in the lexicographical order of the persistent pairs: simplices in a persistent pair are adjacent and arranged according to the filtration value of the first element in the pair, and ties are broken by looking at the second element. As a result, denoting by $A \ni \mathcal{P}$ the set of point clouds in $\mathbb{R}^d$ and $\text{Filt}_\mathcal{K} \subset \mathbb{R}^{|\mathcal{K}|}$

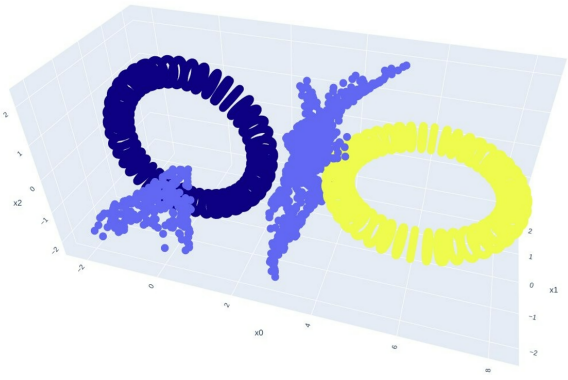


Figure 1: The yellow and dark blue points are the points of a dataset, the light blue points are a sampling of the decision boundary.

the set of vectors in $\mathbb{R}^{|\mathcal{K}|}$ that define a filtration on $\mathcal{K}$, persistent homology can be seen as a composition $\text{Pers} \circ \Phi : A \to \mathbb{R}^{|\mathcal{K}|}$ of a function $\text{Pers} : \text{Filt}_{\mathcal{K}} \to \mathbb{R}^{|\mathcal{K}|}$ with the filtration function Φ. Crucially, $\text{Pers} \circ \Phi$ is differentiable everywhere in A except on a submanifold S, with $\dim S < \dim A$.

The persistence diagram D of a point cloud can be visualized in the form of a set of points in $\mathbb{R}^2$, where for each dimension k, a point (x, y) represents the filtration values of a k-dimensional persistent pair. All the points lie above the diagonal line, since y is the filtration value of a simplex that destroyed a feature in the homology, and must therefore be larger than the filtration value x of the simplex that created the corresponding feature.

One important limitation of persistent homology is the sensibility to the sampling of the manifold: the persistence diagram D of a manifold $\mathcal{M}$ can only carry information about the topology of densely enough sampled regions of $\mathcal{M}$. Moreover, regions of the manifold with a more complex topology (e.g., several small connected components) require a denser sampling for correctly retrieving their homology (see Hickok 2021).

1.3 Problem setting

Common methods to reduce overfitting consist in reducing the expressive power of the model. This is done by imposing constraints on the wights, e.g. on the l_2 norm of the weight vector in weight decay, or on the sharpness of the minimum of the loss function in SAM. These methods are blind to the implication of a given weight in overfitting, as the constraints are uniformly applied to all the weights. In contrast to that, regularization methods should strive to target specifically the weights that cause overfitting. Moreover, in applications the decision boundary has additional connected components that are not useful for the classification task, and make the model vulnerable to adversarial attacks (as illustrated in fig. 1, where a useless connected component of the decision boundary can be seen close to the blue torus). Being able to recognize and delete these connected components would allow for a more robust training.

Our proposition is to enforce homological constraints on the decision boundary of the network f_w. This can help avoid overfitting by imposing the simplest possible topology that still allows to fit the training set. Moreover, if the topology of the dataset is known before training (e.g. thanks to the tools proposed in Ramamurthy, Varshney, and Mody 2018), this provides the tools for introducing the corresponding inductive bias on the network. In particular, the homology is optimized through a topological loss function, a real-valued function L_T that encapsulates the desired constraints.

2 Method

2.1 Topological losses

Persistence diagrams provide a powerful tool to interpret the homology of a manifold. For the Vietoris-Rips filtration, the points of H_0, which correspond to 0-dimensional cycles (i.e., distinct connected components) have their first coordinate equal to 0, since all the connected component appear at $\epsilon = 0$. The second coordinate is the filtration value of the edge that merged two connected component. For k-dimensional homology, the first coordinate is the filtration value of the simplex that closed a k-dimensional cycle, while the second coordinate is the filtration value of the simplex that filled the same cycle.

The ease of interpretation of persistence diagrams allows to define easily interpretable functions of the homology: for instance, the loss function $L_t = \sum_{(x,y)\in D_1}(y-x)^2$ penalizes the lifetime of 1 dimensional cycles (i.e., circles), so optimizing it causes circles to shrink. However, one should still keep in mind that the persistence diagram cannot be entirely explained with the homology of the underlying manifold, some of the carried information comes from the sampling. Therefore, the fraction of the loss that corresponds to the manifold's homology depends on the density of the sampling.

The next step is to allow the gradient of the topological loss to propagate back not only to the points sampled from the decision boundary, but to the weights of the network that parameterizes it.

2.2 Gradient of the topological loss

The sampling of the decision boundary is obtained as follows: We first draw a set $\mathcal{P}$ from the uniform distribution on the input space. We then optimize the squared difference of the logits

$$L_P = \sum_{p\in\mathcal{P}}(f_w(p)_1 - f_w(p)_2)^2 \tag{1}$$

to push the points to the decision boundary. For each point p, we compute the normal vector $n(p)$ of the decision boundary evaluated at p by evaluating (and normalizing) the gradient of the difference of the logits at each point. We then get a parametrization of a new set of points $\mathcal{P}' = \{p'_i = p_i + t_i n(p_i) \quad \forall\, p_i \in \mathcal{P}\}$, with $T = (t_i)_{i=1,\dots,|\mathcal{P}|} \in \mathbb{R}^{|\mathcal{P}|}$ the vector with entries parameterizing the positions of points in $\mathcal{P}'$ (see appendix A). We define an optimality condition

$$Opt\colon \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n : T \times w \mapsto (f_w(p_i + t_i n(p_i))_1 - f_w(p_i + t_i n(p_i))_2)_{i\in\{1,\dots,n\}} \tag{2}$$

The optimality condition implicitly defines the position of the points in $\mathcal{P}'$ along the normal lines of the decision boundary: $Opt(T^\star(w), w) = 0$. The optimality condition is satisfied for $T = 0$, that is, when the position of $\mathcal{P}'$ matches that of $\mathcal{P}$. Moreover, as $Opt(T^\star(w), w)$ is a constant function of w, its gradient is 0. This allows to compute the gradient of $T^\star$ using the chain rule

$$\begin{aligned} \frac{\partial}{\partial w}Opt(T^\star(w), w) = \frac{\partial}{\partial T}Opt \cdot \frac{\partial}{\partial w}T^\star + \frac{\partial}{\partial w}Opt = 0 \\ \Rightarrow \frac{\partial}{\partial w}T^\star = -\left(\frac{\partial}{\partial w}Opt\right)^{-1} \cdot \frac{\partial}{\partial w}Opt \end{aligned} \tag{3}$$

This linear system is automatically solved by jaxopt (Blondel et al. 2021). The variables $\mathcal{P}'$ were introduced so there is a single degree of freedom per point, which makes $\partial Opt/\partial w$ full rank.
The computed term $\frac{\partial}{\partial w}T^\star$ can be used in the computation of the gradient of the topological loss :

$$\frac{\partial}{\partial w}L_T = \frac{\partial}{\partial T^\star}L_T \cdot \frac{\partial}{\partial w}T^\star \tag{4}$$

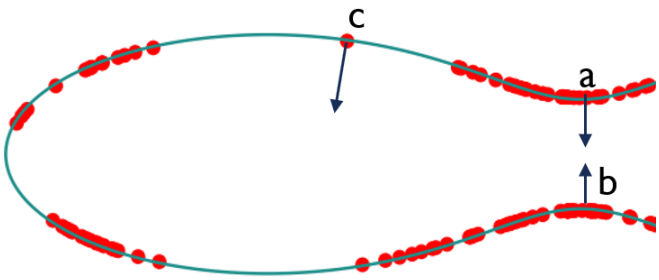


Figure 2: Points a and b are closer together than a and c according to euclidean distance, but not according to the intrinsic distance. A better estimation of the intrinsic distance can be found by taking into account that the normal vectors of a and c are similar, contrary to b.

2.3 Normal metric filtration

In the general setting of persistent homology, the only available information about the manifold $\mathcal{M}$ is the sampling $\mathcal{P}$. In contrast, our setting allows to get additional information in the form of the normal unit vectors n_i of $\mathcal{M}$ evaluated at the points $\mathcal{P}$. The normal vectors are given by first computing the gradient of the difference of the logits :

$$n_i' = \frac{\partial}{\partial p_i}\Big(\sum_{p_i \in \mathcal{P}} f_w(p_i)_1 - f_w(p_i)_2\Big)(\mathcal{P}) \tag{5}$$

The vectors n_i are obtained by normalizing n_i'.

Consider two points $p_1, p_2 \in \mathcal{P}$, and their normal vectors n_1 and n_2. In the usual setting of persistent homology, the filtration value $\Phi(p_1, p_2)$ of the vertex (p_1, p_2) is the euclidean distance $\|p_1 - p_2\|$. However, in order to resemble the underlying manifold $\mathcal{M}$, it is important that $\Phi(p_1, p_2)$ is close to the intrinsic distance $d_{\mathcal{M}}(p_1, p_2)$, which is not necessarily the case of euclidean distance (as illustrated in figure 2).

To that end, the normal unit vectors provide useful information: if $\|n_1 - n_2\|$ is large, then the manifold must be curved between p_1 and p_2. This motivates the following definition :

Definition 2.1 (Normal metric filtration)**.** Let $\mathcal{P}$ be a subset of points in $\mathbb{R}^d$, $\mathcal{N}$ the set of normal unit vectors of $\mathcal{M}$ evaluated at $\mathcal{P}$ and ϵ a real positive value. The simplicial complex $\mathcal{K}_\epsilon$ is the minimal simplicial complex such that for each simplex $\sigma \in \mathcal{K}_\epsilon$ we have

$$\sqrt{\|p_i - p_j\|^2 + \lambda\|n_i - n_j\|^2} < 2\epsilon \quad \forall\, p_i, p_j \in \mathcal{V}_\sigma$$

where $n_i \in \mathcal{N}$ is the normal unit vector evaluated at p_i.

In essence, we capture the additional information by embedding $\mathcal{P}$ in a higher dimensional space using the concatenation map $\mathbb{R}^d \to \mathbb{R}^{2d}$: $p_i \mapsto (p_i, \lambda n(p_i))$. The obtained embedding $\mathcal{P}_n$ captures the additional information about intrinsic distance carried by the normal vectors. We then compute the persistent homology of $\mathcal{P}_n$.

2.4 TopOpt

Here, we describe `TopOpt`, our algorithm for optimizing the homology of the decision boundary of a neural network. The algorithm receives a neural network f depending on a set of parameters w, as

well as the dataset of samples $(x_i, y_i)_{i\in|\mathscr{D}|}$ with labels y_i. `TopOpt` can also work on neural networks with a single output, in which case the decision boundary is considered to be the set of points where the output of the network is 0. The pseudocode for the algorithm is given in 1.

Algorithm 1 TopOpt

Input : network f, parameters w, data points X, labels Y, sampling epochs p, learning rate α, optimization epochs m, cross entropy relative importance c

$\mathcal{P} \leftarrow$ uniform sampling the latent space
for $i = 1$ to p **do**
  $\mathcal{P} \leftarrow \mathcal{P} - \alpha\nabla L_P(\mathcal{P})$ ▷ sample the decision boundary using equation (1)
end for
for $j = 1$ to $\mid \mathcal{P} \mid$ **do**
  $n'_j \leftarrow \frac{\partial}{\partial p_j}\Big(\sum_{p_j\in\mathcal{P}} f_w(p_j)_1 - f_w(p_j)_2\Big)(\mathcal{P})$
  $n_j \leftarrow \frac{n'_j}{|n'_j|}$ ▷ compute the normal unit vectors
end for
for $i = 1$ to m **do**
  $\hat{y_1}, \hat{y_2} \leftarrow f_w(X)_1, f_w(X)_2$
  $\hat{p_1}, \hat{p_2} \leftarrow \text{Softmax}(\hat{y_1}, \hat{y_2})$
  $\frac{\partial}{\partial w}T^\star \leftarrow -\left(\frac{\partial}{\partial w}Opt\right)^{-1} \cdot \frac{\partial}{\partial w}Opt$ ▷ get the implicit gradient using Opt defined in (3)
  $L(w) = L_T(w) - c \cdot (y\log(p_1) + (1-y)\log(p_2))$
  $w = w - \alpha \cdot \nabla L(w)$
  $\mathcal{P} \leftarrow \mathcal{P} - \alpha\nabla L_P$ ▷ update the position of the sampled points
end for

3 Experimental Setup

To demonstrate our results, we first use a toy model

$$f_w(x, y) = \exp(-1.5((x-1)^2 + y^2)) + \exp(-1.5((x+1)^2 + y^2)) - (w + 0.75)$$

that models a single output network with a single parameter w that takes a $2-$dimensional input (x, y), or a sum of Gaussian kernels whose trainable parameters are the positions of the kernels' centers. We then create the dataset shown in Fig.5a, and train a multilayer perceptron with 5 hidden layers of 10 neurons and ReLU activation on it. We then observe the decision boundary before and after the topological optimization, as well as the persistence diagrams.

4 Results and discussion

In the example in figure 3, the decision boundary is the $0-$level set of f_w, which is a sum of two Gaussian kernels, while w is added to the output to control the elevation of the graph. Hence, the $0-$level set can consist of two distinct cycles, or a single one, depending on the value of w. In figure 3, we start with two cycles and optimize w with a topological loss that penalizes any connected component or cycle except for the largest one, and incentives a single large cycle. The effect is a reduction of w from 0 to ≈ -0.39, which allows to merge the cycles.

Figure 4 shows the evolution of the $0-$level set of a sum of 6 Gaussian kernels when optimized with the loss function that penalizes connected components. This example illustrates the importance of the normal metric filtration, as the positions of the points would not allow to distinguish the connected components. Moreover, on example we can observe that the connected components merge one after the other. This is a consequence of the implementation of the loss function, that

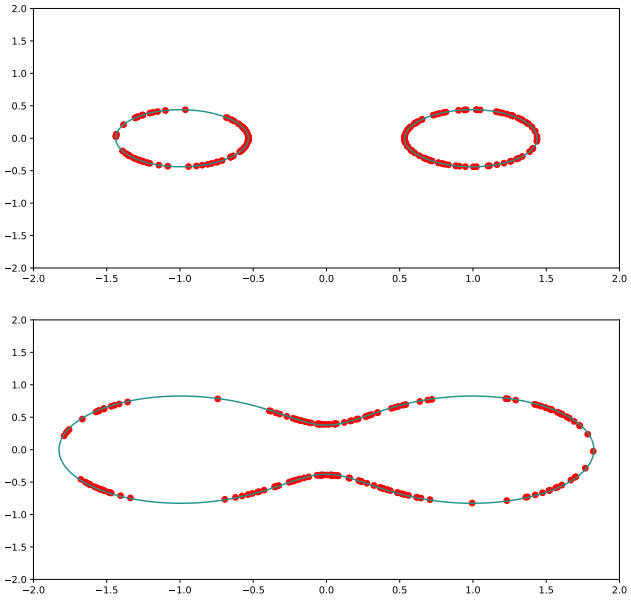


Figure 3: Top : the $0-$level set of f_w before the optimization, with $w = 0$. Bottom : the $0-$level set of f_w after the optimization with the 'single_cycle_and_connected_component' loss, with $w \approx -0.39$. The red points are the sampling of the $0-$level set.

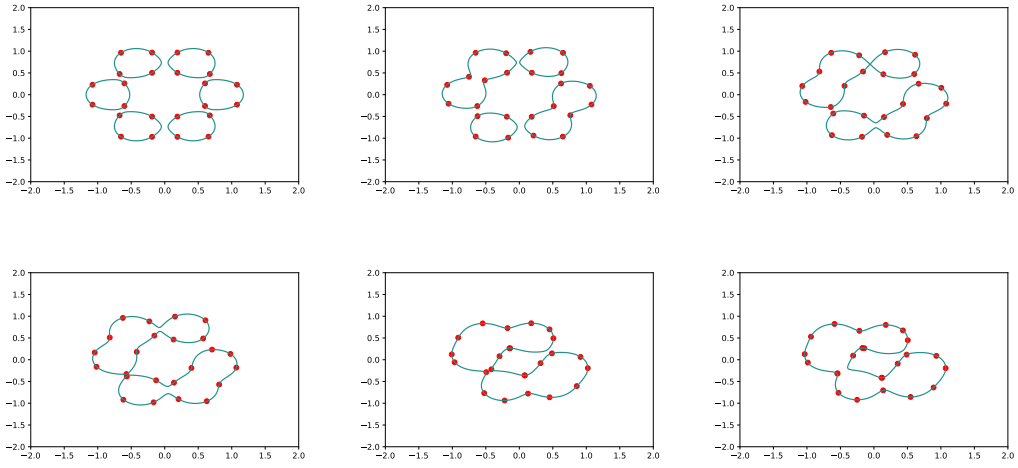
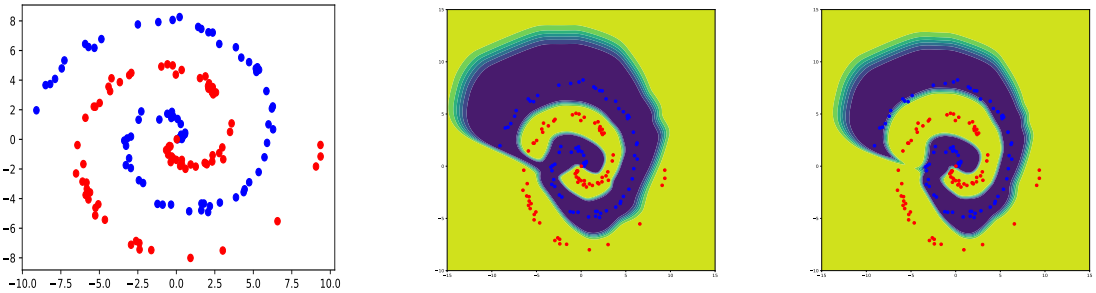


Figure 4: $0-$level set of a sum of 6 gaussian kernels optimized with a loss that penalizes any connected component but the largest.



(a) A 2-dimensional dataset with 2 classes.

(b) The decision boundary of the network before optimization.

(c) The decision boundary of the network after optimization.

Figure 5: Optimization of the decision boundary of a network trained on the spiral dataset.

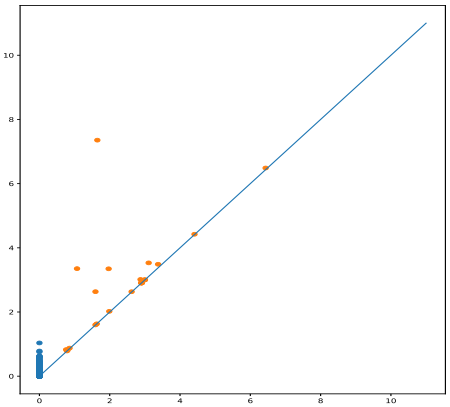
only penalizes the distance between the two most distant connected components.

In Fig.5 we can observe that although the network achieves 100% training accuracy, the shape of the dataset is not correctly retrieved. After applying `TopOpt` with the 'Single_connected_component' loss we obtain the decision boundary in 5c. We can observe that the 'bridge' connecting the inside of the blue spiral to its center was cut, and the obtained decision boundary has the desired single cycle and connected component.

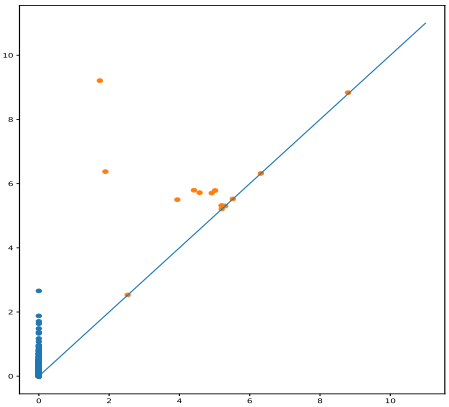
Figure 6 displays the persistence diagrams before and after the optimization. We can observe that the persistence diagrams of the normal metric filtration better captures the homology of the decision boundary: H_1 clearly displays two cycles before the optimization, which the normal persistence diagram fails to do. The largest (finite) lifetime displayed in H_0 is clearly higher than the rest of the features in H_0 before the optimisation, which indicates that the last merge of connected components happens long after the other merges. This corresponds to the two connected components in the decision boundary in Fig 5b. After the optimization, the persistence diagram of the normal metric filtration shows that all the features in H_0 are grouped together, which is the consequence of the choice of the loss function that penalized the filtration value of the last merge.

5 Constraining implicit representations of 3D objects as level sets of a neural network

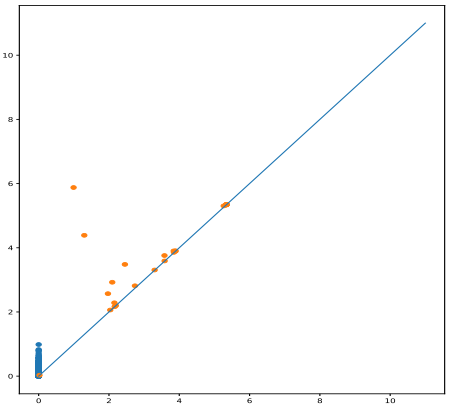
In Vicini, Speierer, and Jakob 2022, the authors propose a method for Physically-based differentiable rendering, that aims to represent 3D objects as level sets of neural networks. This is done by optimizing a loss function that describes the difference between the physical object and the level set. Here, `TopOpt` can be used to enforce topological constraints on the 3D representation. This can be useful to avoid artifacts such as additional connected components. We illustrate in figure 7 how the number of connected components can be reduced using `TopOpt`.



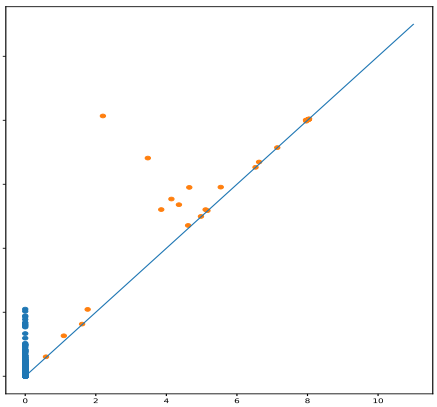
(a) The persistence diagram before the optimization.



(b) The persistence diagram of the normal metric filtration before the optimization.

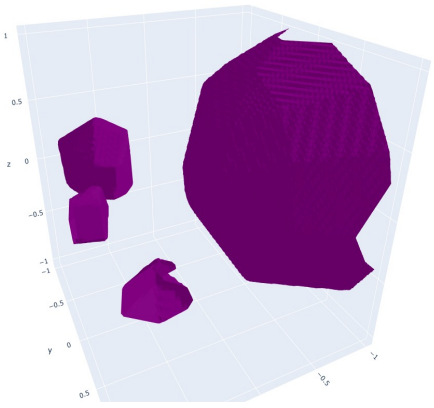


(c) The persistence diagram after the optimization.

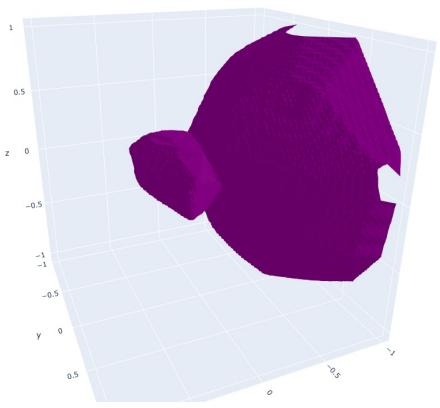


(d) The persistence diagram of the normal metric filtration after the optimization.

Figure 6: The persistence diagrams of the decision boundary before and after the optimization. The dots correspond to H_0, and the orange dots to H_1. The diagrams only show finite valued persistence pairs, so the highest dot in H_0 corresponds to the filtration value of the simplex that merges the last two connected components.



(a) The 0-level set of a neural network



(b) The 0-level set after the optimization.

Figure 7: Optimization of the level set of a network with a 3 dimensional input.

Conclusion

In this article we introduced a framework that allows to optimize functions of the topology of the decision boundary. In particular, our algorithm builds on Carrière et al. 2020 to optimize manifolds given as levelsets of neural networks.

Future work might allow to e.g. optimize the curvature of the decision boundary. The motivation for such work resides in the fact that parts of the decision boundary with high absolute principal curvature can correspond to overfitting, as it would allow to fit a noisy dataset. Moreover, additional work can allow overcoming the issue of utilizing our method with high dimensional datasets by sampling only the relevant parts of the decision boundary (e.g., around misclassified training samples).

References

[Blo+21] Mathieu Blondel et al. "Efficient and Modular Implicit Differentiation". In: *CoRR* abs/2105.15183 (2021). arXiv: 2105.15183. URL: https://arxiv.org/abs/2105.15183.

[Car+20] Mathieu Carrière et al. "A note on stochastic subgradient descent for persistence-based functionals: convergence and practical aspects". In: *CoRR* abs/2010.08356 (2020). arXiv: 2010.08356. URL: https://arxiv.org/abs/2010.08356.

[For+20] Pierre Foret et al. "Sharpness-Aware Minimization for Efficiently Improving Generalization". In: *CoRR* abs/2010.01412 (2020). arXiv: 2010.01412. URL: https://arxiv.org/abs/2010.01412.

[Hic21] Abigail Hickok. "A Family of Density-Scaled Filtered Complexes". In: *CoRR* abs/2112.03334 (2021). arXiv: 2112.03334. URL: https://arxiv.org/abs/2112.03334.

[Kwo+21] Jungmin Kwon et al. "ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks". In: *CoRR* abs/2102.11600 (2021). arXiv: 2102.11600. URL: https://arxiv.org/abs/2102.11600.

[RVM18] Karthikeyan Natesan Ramamurthy, Kush R. Varshney, and Krishnan Mody. *Topological Data Analysis of Decision Boundaries with Application to Model Selection*. 2018. DOI: 10.48550/ARXIV.1805.09949. URL: https://arxiv.org/abs/1805.09949.

[VSJ22] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. "Differentiable Signed Distance Function Rendering". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 41.4 (July 2022), 125:1–125:18. DOI: 10.1145/3528223.3530139.

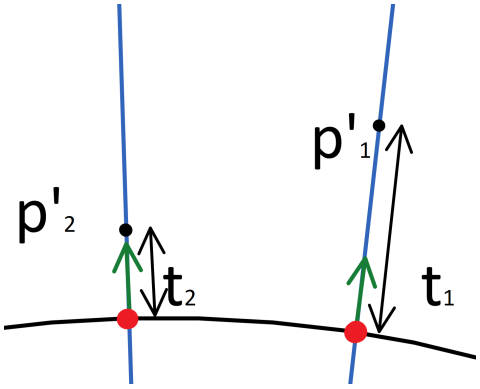


Figure 8: points in $\mathcal{P}'$ are parametrized by a single real value t_i that determines the position of p'_i along the normal line of the decision boundary that passes through p_i.

Appendices

A Implicit definition of the decision boundary

In section 2.2, we explained how to define a new set $\mathcal{P}'$ of points that lie on the normal line of the decision boundary that passes through the corresponding points in $\mathcal{P}$ (see fig. 8). As a result, there is a single degree of freedom for each variable of the optimality condition. This is necessary to make the linear system solved by jaxopt solvable and well posed. Indeed, if the points were free to move along the decision boundary, the matrix inverted by jaxopt would not be full rank.