UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

ADVERSARY DETECTION IN NEURAL NETWORKS VIA PERSISTENT HOMOLOGY

Autor: Severino Da Dalt

Director: Dr. Francisco Belchí Guillamón

Realitzat a: Department of Mathematics and Computer Science

Barcelona, 24 de gener de 2021

Contents

Abstract

The main goal of this work is to present a recently-invented homology theory called *persistent homology* and its application on the detection of *adversary examples* of neural network presented in the paper [4].

2010 Mathematics Subject Classification. 55NXX, 55N35, 55-04, 62-07

Acknowledgement

First, I would like to express my sincere gratitude to Dr. Francisco BelchÃ, who accepted tutoring me. This work wouldn't have been possible without his help and advice.

Besides my tutor, I would like to thank all my friends that accompanied me through my studies, and made of these years the best period of my life.

I thank Fritz, Jorge and Pau, my roommates, who supported me since the first time I met them.

Finally, special thanks to my parents, who, with their help and appreciation, motivated me to reach my goals.

Introduction

Persistent homology is a method used in topological data analysis that studies qualitative features of data that persist across multiple scales. This relatively new theory has become particularly popular in these last decades given the many new applications that has been found for it, ranging from biological systems [6] to artificial neural networks [4]. In this work we will focus on the last one, studying a particular application presented by *Thomas Gebhart* and *Paul Schrater* in their work *Adversary Detection in Neural Networks via Persistent Homology* [4].

The work here presented is divided in two segments. In the first chapter, our objective is to define persistence. In order to do this we first define *simplicial homology*, that will set the fundamental theory that we need. Also, we will present *Čech and Rips complexes*, a special kind of simplicial complex that will help the intuitive and graphic understanding of what persistence does (and also will be used in the adversary detection methods). Finally we will define the *persistent homology* and show the robustness of this theory given by the *Structure theorem* and the *Stability theorem*.

In the second part of the work we will focus on the applications of the theory. First, we will need a brief introduction to what a *neural network* and its *adversary examples* are. Then, we will show how a neural network can be represented as a weighted graph and thus, how we can use persistence to see which substructures of the network are the most solid depending on the input. Finally, we will present the 3 methods for adversary detection and show the results of the testing done in [4].

Notation: During this work, we will consider every vector space as a $\mathbb{Z}_2$ vector space ($= \mathbb{Z}/2\mathbb{Z}$ vector space), except when we specify.

Chapter 1

Persistent Homology

Persistent homology is a tool from topological data analysis that detects the global features of a topological space. It has a purpose similar to that of many other homologies, since it counts the "holes" in different dimensions of a space. But instead of just counting those features, it studies their *persitence*, i.e. their lifespan in a space which "evolves" through a parameter ε. In order to study the space in each instant ε, it uses the *simplicial homology groups*, and that is why we introduce them first.

1.1 Simplicial Homology

Let's first define some basic concepts. The standard *n-simplex* is the subset of $\mathbb{R}^{n+1}$ given by: $\Delta^n = \{(t_0, ..., t_n) \in \mathbb{R}^{n+1} | \sum_{i=0}^{n} t_i = 1 \text{ and } t_i \geq 0 \ \forall i\}$. A *Simplicial Complex* is a set of simplices which satisfies the following conditions:

(a) For each simplex δ in the set, the faces of δ are also in the set.

(b) If the intersection of two simplices of the set δ_1 and δ_2 is non-empty, then it is a face of δ_1 and a face of δ_2.

Let $\mathcal{K}$ be a simplicial complex. The $\mathbb{Z}_2$ vector space generated by the p-dimensional simplices of $\mathcal{K}$ is called $C_p(\mathcal{K})$. Its elements are called *p-chains* and they are formal sums $c = \sum_j \gamma_j * \sigma_j$ where γ_j is 0 or 1 and σ_j's are p-simplices of $\mathcal{K}$. We also define a *boundary map* ∂ which pairs a p-simplex to the sum of its (p-1)dimensional faces. This definition can be extended linearly for chains. Consider $c \in C_p(\mathcal{K})$:

$$\partial(c) = \sum_j \gamma_j * \partial(\sigma_j)$$

The algebraic structure formed by the sequence $(C_p(\mathcal{K}))_{p\geq 0}$ together with the boundary map ∂ is called the *chain complex* of $\mathcal{K}$, and we denote it $C_*(\mathcal{K})$.

Notice that, since each $C_p(\mathcal{K})$ is a $\mathbb{Z}_2$ vector space, $\sigma_j + \sigma_j = 2 * \sigma_j = 0$. Also, the composition of ∂ with itself is the zero-map: $\partial \circ \partial = \partial^2 = 0$ (the proof of this can be found at page 105 of [1] and works mutatis mutandis changing $\mathbb{Z}$ into $\mathbb{Z}_2$).

The *p-cycles* are those p-chains c such that their image through the boundary map is 0: $c \in ker(\partial)$. The *p-boundaries* are those p-chains c which are the image of a (p+1)-chain through the boundary map: $c \in Im(\partial)$. Finally we can define the simplicial homology.

Definition 1.1. The *p-th simplicial homology group* is defined as the quocient of the subspace of C_p formed by all the p-cycles (called Z_p) and the subspace of C_p formed by all the p-boundaries (called B_p):

$$H_p(\mathcal{K}) = Z_p / B_p$$

The ranks of these groups are called the *Betti numbers* of $\mathcal{K}$:

$$\beta_p = dim_{\mathbb{Z}_2} H_p(\mathcal{K})$$

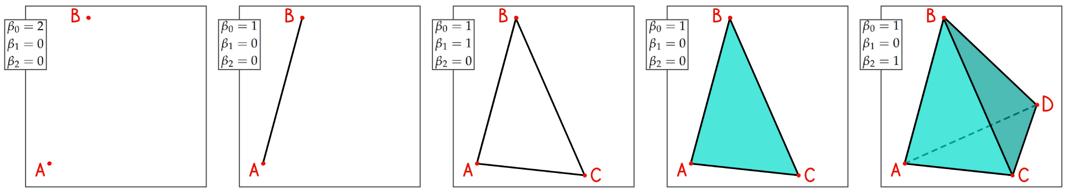

Figure 1.1: This image shows 5 different simplicial complexes and their respective Betti numbers β_0, β_1 and β_2.

Now that we have introduced simplicial homology, we can start explaining the idea of persistence. Consider a topological space M. The first step is to construct a *filtration* of M, which we define as a set of subspaces of M, $\{M_i\}_0^m$ such that $\forall i \in \{0, ..., m-1\}$, $M_i \subset M_{i+1}$. Although this filtration can be constructed with any kind of topological space, we will introduce it for Čech and Rips complexes, since it is what we will use to study neural networks (in particular we will use Rips complexes). Also, their geometric nature helps understand persistent homology in a more intuitive and graphical way.

1.2 Čech and Rips complexes

Consider we have a collection of points $\{x_m\}$ in an Euclidean n-dimensional space $\mathbb{E}^n$ representing a lower dimensional space. Since we want to study the features of the object, we can build and study a simplicial complex considering that the $\{x_m\}$ are the vertices (0-simplices) of the complex, and connecting through an edge (1-simplex) each pair of points which fulfills a proximity condition, for example being at a distance smaller than a fixed ε. Then use the structure formed by those vertices and edges as a scaffold for higher dimensional simplices. The choice on how to fill those higher dimensional simplices in the complex allows for different global representations. Two common methods to complete the simplicial complex are:

Definition 1.2. Given a collection of points $\{x_m\}$ in an Euclidean space $\mathbb{E}^n$, the *Čech complex*, $\mathcal{C}_\varepsilon$, is the abstract simplicial complex whose k-simplices are determined by unordered (k+1)-tuples of points $\{x_m\}_0^k$ whose closed $\varepsilon/2$-ball neighborhoods have a point of common intersection.

Definition 1.3. Given a collection of points $\{x_m\}$ in a Euclidean space $\mathbb{E}^n$, the *Rips complex*, $\mathcal{R}_\varepsilon$, is the abstract simplicial complex whose k-simplices correspond to unordered (k+1)-tuples of points $\{x_m\}_0^k$ that are pairwise within distance ε.

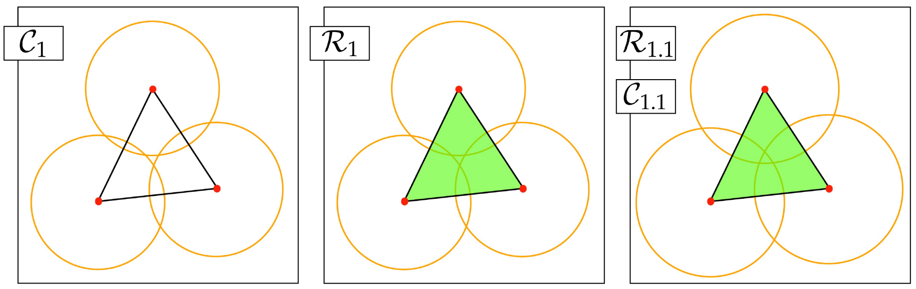

Figure 1.2: We consider 3 points in $\mathbb{E}^n$. For $\varepsilon = 1$, since each pair of points stands within a distance smaller then 1, all 3 edges are added to both the Čech and Rips Complex, but since the 3 $\varepsilon/2$-balls have no point in common, the 2-simplex only is added to the Rips complex (on the left $\mathcal{C}_1$, in the middle $\mathcal{R}_1$). If we choose $\varepsilon = 1.1$ instead, the 2-simplex is added to both complexes (on the right, $\mathcal{C}_{1.1} = \mathcal{R}_{1.1}$).

Let's talk about the main differences between these two complexes.

It follows the definitions that the Čech complex $\mathcal{C}_\varepsilon$ is included in the Rips complex $\mathcal{R}_\varepsilon$, $\mathcal{C}_\varepsilon \subset \mathcal{R}_\varepsilon$, since the condition of having k points whose closed $\varepsilon/2$-ball neighborhoods have a point of common intersection is a stronger condition then having those points pairwise within distance ε (see Figure 1.2).

Another important difference between the complexes is shown by the Čech theorem, which states that $\mathcal{C}_\varepsilon$ has the homotopy type of the union of closed radius $\varepsilon/2$ n-dimensional balls centered on the points $\{x_m\}$. That is why, though the definition might suggest it can be an abstract simplicial complex of higher dimension

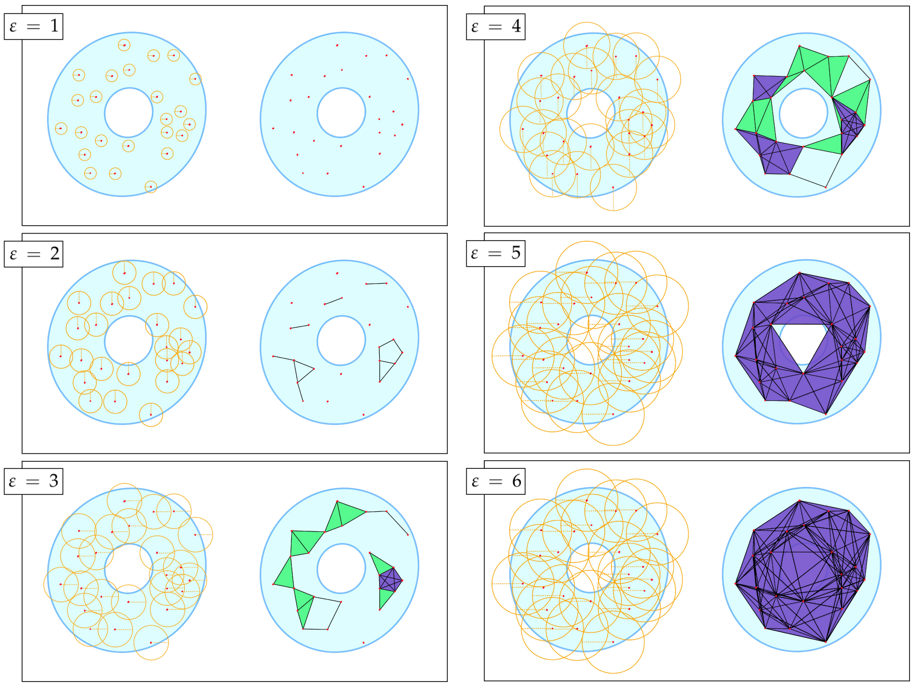

Figure 1.3: A filtration of Rips complexes $\mathcal{R}_\varepsilon$ for a finite set of points representing a disc with a hole. As we can see in the image, holes appear and disappear in the complex while we increase the value of ε. The objective of persistence is to distinguish between the real hole in the middle and others which are just noise.

then n, it behaves exactly like a subspace of $\mathbb{E}^n$. This makes Čech complexes much more reliable, in the meaning of describing the features of the object, than Rips complexes, which in general don't necessarily behave like an n-dimensional space at all (see Figure 1.4).

On the other hand, Rips complexes are less expensive, computationally speaking, then Čech's, despite the first having more simplices. The fact is that the Rips complex is a *flag complex*, i.e., it is maximal among all simplicial complexes with the same 1-simplices structure. This way, the 1-skeleton (all the 1-simplices) fully determines the complex, and it can be stored as a graph. Instead, the Čech complex requires the entire boundary relation between simplices to be stored.

That is why, as we will see, persistent homology is usually calculated for Rips complexes in order to deduct (using Lemma 1.6) the homology of the Čech complexes.

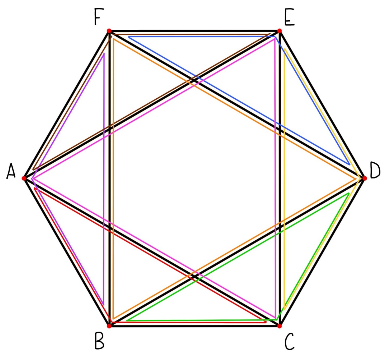

Figure 1.4: Consider the vertices A,B,C,D,E,F of a regular hexagon with edge length = 1. Then, use those 6 points as the vertices of a Rips complex of constant $\varepsilon = 1,8$. The resulting complex has only 8 2-simplices, the triangles: ABF (purple), BCD (green), DEF (blue), ACE (pink), ABC (red), CDE (yellow), AFE (brown) and BDF (orange). The union of the first 4 2-simplices fills the whole hexagon and so do the last 4. Hence, we can think of each of these unions as an hemisphere of a sphere. Plus, there are no 3-simplices since no subset of 4 vertices has all pairs of vertices within the distance 1,8. Then, $\mathcal{R}_\varepsilon$ is homotopy equivalent to a sphere, which cannot be embed in a plane.

1.3 Persistence

Now that we have presented a way to construct the complex, a natural question may be which is the optimal ε to do so. If we choose an ε too small, some cycles may show while others do not, since they require bigger simplices to be added to the complex. But even if we take an ε large enough in order to show those "bigger" cycles, the smaller ones might get trivialized in a higher-dimensional simplex (see Figure 1.3). The idea is that a "perfect" ε might not exist, so asking for an optimal value, makes no sense. A substitute for this question might be: which are those features which *persist* the longer varying ε? And in order to answer this, we use persistent homology.

Consider a filtration of Rips complexes $\{\mathcal{R}_i\}_1^N$ and for each $\mathcal{R}_i$ consider its chain complex $C_*(\mathcal{R}_i)$ (which we will denote C_*^i in order to simplify the notation). We now have a sequence of chain complexes $C = (C_*^i)$ together with the induced chain maps $x : C_*^i \to C_*^{i+1}$ (We do not index the maps for notation reasons). We call this algebraic structure a *persistence complex*. Note that the maps x are inclusions in our case, being that the inclusion $\mathcal{R}_i \subset \mathcal{R}_{i+1}$ implies $C_*^i \subset C_*^{i+1}$.

Now, our goal is to see how homological classes persist throughout the filtration. For this purpose we consider the maps:

$$f_p^{i,j} : H_p(C_*^i) \to H_p(C_*^j)$$

which are induced by the inclusions $C_*^i \subset C_*^j$, $\forall i < j$.

We then say that a class α is *born* at $\mathcal{R}_i$ if $\alpha \in H_p(C_*^i)$ and it is not in the image of $f_p^{i-1,i} : H_p(C_*^{i-1}) \to H_p(C_*^i)$. If α is born at $\mathcal{R}_i$, we say that it *dies entering* $\mathcal{R}_j$ if the image of the map $f_p^{i-1,j-1} : H_p(C_*^{i-1}) \to H_p(C_*^{j-1})$ does not contain the image α, but the image of $f_p^{i-1,j} : H_p(C_*^{i-1}) \to H_p(C_*^j)$ does. Since α is born at the instant ε_i and dies entering the instant ε_j, we say that the *persistence* of α is $\varepsilon_j - \varepsilon_i$. We can finally define the persistent homology.

Definition 1.4. For $i < j$, the *p-th (i,j)-persistent homology group* $H_p^{i \to j}(C)$ is defined as the image of $f_p^{i,j} : H_p(C_*^i) \to H_p(C_*^j)$, the map induced by the inclusion $C_*^i \subset C_*^j$:

$$H_p^{i \to j}(C) = im(f_p^{i,j})$$

The rank of the image is called *p-th (i,j)-persistent Betti number* of f:

$$\beta_p^{i,j} = rank\ im(f_p^{i,j})$$

Intuitively, $\beta_p^{i,j}$ is the number of p-dimensional classes that are born before $\mathcal{R}_i$ and are still a alive in $\mathcal{R}_j$. In particular, if $i = j$, we have that $H_p^{i \to j}(C) = H_p(C_*^i)$ and $\beta_p^{i,j} = \beta_p$, since the classes in $H_p(C_*^i)$ are born before $\mathcal{R}_i$ and dies after $\mathcal{R}_i$.

Now that we have defined persistence, a natural question might be: is the classification of how classes are born and die unique? (see Figure 1.5)
The answer is affirmative and it's a direct consequence of theorem 1.5 which we will introduce shortly.

Consider $R[x]$ the ring of polynomials on a commutative ring R. If we compute homology with coefficients in the ring R, we can give a graded $R[x]$-module structure to a persistence module $C = (C_*^i)$ with $x : C_*^i \to C_*^{i+1}$ acting as a shift map, i.e. the monomials $x^n \in R[x]$ (we refer to x^n as the composition of n maps x) send classes from C_*^i to C_*^{i+1}. We will assume that the C_*^i's are finitely generated.

Then, C is free as a $R[x]$-module but, the resulting homology $H_*(C, F) = \bigoplus_{i=0}^N H_*(C_*^i, F)$ (We are assuming that for a given $N > 0$, $H_p(C_*^i, F) = 0, \forall p > N$, $\forall i$) is not necessarily free, nor we need it to be so. The problem stands in the fact that $R[x]$-module are generally very difficult to classify.

If instead we choose a field F, for example $\mathbb{Z}_2$ (that's why we defined persistence with it), we can use the decomposition theorem of finitely generated modules over PID's, since the only graded ideals of $F[x]$ are $x^n F[x]$ for $n > 0$. The consequence of this are resumed in thefollowing theorem.

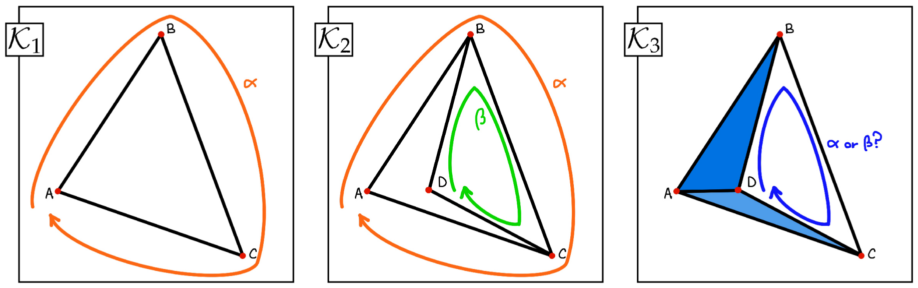

Figure 1.5: Consider the filtration of simplicial complexes shown in the image $\mathcal{K}_1 \subset \mathcal{K}_2 \subset \mathcal{K}_3$. $\mathcal{K}_1$ is an empty triangle formed by the three 1-simplices AB, BC and CA (on the left). $\mathcal{K}_2$ carries the triangle from $\mathcal{K}_1$ and adds two 1-simplices BD and CD (in the middle). $\mathcal{K}_3$ adds a 1-simplex AD and 2 2-simplices ACD and ABD to the previous complex (on the right). It's easy to see that $H_1(\mathcal{K}_1)$ only has one generator α and $H_1(\mathcal{K}_2)$ has two, α and β. $H_1(\mathcal{K}_3)$ has one generator, but how should we call it? α or β? Our intuition say that it should be α since it's the elder class, and theorem 1.5 proves that this is the one that respects the algebraic structure.

Theorem 1.5. *For a finite persistence module C with field F coefficients,*

$$H_*(C,F) \cong \bigoplus_i x^{t_i} * F[x] \oplus \left(\bigoplus_j x^{r_j} * \left(F[x] / \left(x^{s_j} * F[x] \right) \right) \right)$$

In other words, $H_*(C,F)$ decomposes in a free portion and a torsional portion *uniquely*.

The free portion represents those homology classes which are born at the instant t_i and persist through all the filtration. On the other hand, the torsional portion corresponds to those homology classes which are born at the instant r_j and dies at $r_j + s_j$.

As a last remark for this section, we present the following lemma, which is a useful tool that helps us relate the persistent homology groups of Čech and Rips complexes.

Lemma 1.6. *Consider a collection of points $\{x_m\}$ in an Euclidean space $\mathbb{E}^n$, and $\mathcal{R}_\varepsilon$, $\mathcal{C}_\varepsilon$ the Čech and Rips complexes for $\{x_m\}$. For any > 0 there is a chain of inclusion maps:*

$$\mathcal{R}_\varepsilon \hookrightarrow \mathcal{C}_{\varepsilon\sqrt{2}} \hookrightarrow \mathcal{R}_{\varepsilon\sqrt{2}}$$

This implies that, for $\varepsilon, \varepsilon' > 0$ such that $\varepsilon'/\varepsilon \geq \sqrt{2}$, if a feature of the object persists under the inclusion $\mathcal{R}_\varepsilon \hookrightarrow \mathcal{R}_{\varepsilon'}$, then it is in fact a feature of the Čech complex $\mathcal{C}_{\varepsilon'}$. This way we can approximate the features of a Čexh complex with pairs of Rips complexes. This is very useful since, as we have seen in the previous section, Rips complexes are easier to compute while Čech complexes gives us a more reliable description of the features of the object.

In the following section, we will introduce an intuitive way of representing graphically the persistence of classes: the *barcodes*.

1.4 Barcodes

Let $\{\mathcal{R}_i\}_i$ be a filtration of Rips complexes. Now, consider a plane which horitzontal axis corresponds to the parameter ε, and the vertical axis represents an (arbitrary) ordering of the homology generators of the filtration complexes. For every class of the persistent homology, with birth in $\mathcal{R}_i$ and death at $\mathcal{R}_j$, we draw an horizontal segment with starting point on ε_i and ending point on ε_j. The collection of those horitzontal segments is called *barcode*. When we analyze it with more detail, we see that a barcode holds all the information about the persistent homology of the space we are studying: we know when a class is born (the starting point x-axis value), when it dies (the final point x-axis value) and its persistence (the length of the segment). Also, as the following theorem states, we can easily calculate the persistent homology groups of the space and its correspondent Betti numbers $\beta_p^{i,j}$.

Theorem 1.7. *The dimension of the persistent homology group $H_p^{i\to j}(\mathcal{R})$ is equal to the number of intervals in the barcode of $H_p(\mathcal{R})$ spanning the index interval [i,j]. In particular $H_p(\mathcal{R}_i)$ is equal to the number of intervals which contain i.*

Proof. This is a consequence of the theorem 1.5. Each barcode with finite death instant represent a torsional element in the formula of the theorem while the barcodes that persist until the end of the filtration represent an element of the free portion. □

In figure 1.6 we see the representation of the barcodes for the filtration shown in figure 1.3.

1.5 Morse functions

Although Čech and Rips complexes are what we will end up using in this work, in the this section we present Morse functions in order to generalize the

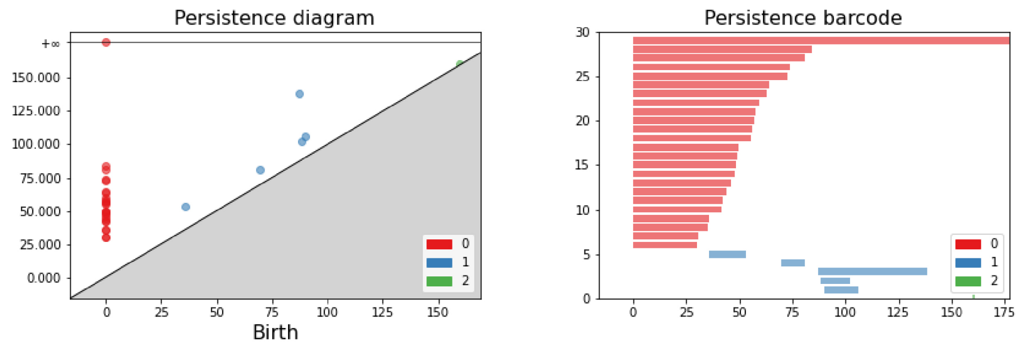

Figure 1.6: This image shows the persistence diagram (on the left) and the persistence barcode (on the right) of the Rips complexes filtration shown in figure 1.3. It has been computed using a software created by Fritz Pere Nobbe [7]. From this diagrams we can deduce that the studied object has one connected component, since only one class of H_0 stand out from the others (represented by the farthest red point from the diagonal int he persistence diagram, and the longest red barcode in the persistence barcode diagram). Also, for the same reasons in H_1, we can deduce it has only one cycle.

use of persistent homology. Also, the subspaces induced by those functions make the explanation of other concepts, like the *persistent diagrams* or the *stability* of persistence, easier and more clear. Note that the filtration we defined for Čech and Rips complexes can be filled with any kind of topological space, and the definition of persistent homology holds mutatis mutandis.

Let's first remember some basic concepts. Consider f a smooth function. We say that x is a *critical point* of f if the differential of f in x is zero, and $f(x)$ is its correspondent *critical value*. A critical point is *non-degenerate* if the Hessian matrix of second partial derivatives is non-singular. Now we can define a Morse function.

Definition 1.8. Let $\mathcal{M}$ be a smooth manifold of dimension d. A smooth function $f : \mathcal{M} \to \mathbb{R}$ is a *Morse function* if it has only non-degenerate critical points all of which have distinct critical values. (Note that, although it takes a choice of coordinates to define the Hessian matrix, the non-singularity is independent of the choice).

Let $f : \mathcal{M} \to \mathbb{R}$ be a Morse function, and we choose regular values $\{t_i\}_{i=0}^{m}$ such that $\forall i \in \{0, ..., m-1\}$ $t_i < p_i < t_{i+1}$ where $\{p_i\}_{i=0}^{m-1}$ are the m critical values of f. For each $t_j \in \{t_i\}_{i=0}^{m}$ we consider the *sublevel set* $\mathcal{M}_j = f^{-1}(-\infty, t_j] \subset \mathcal{M}$ which contains the first j critical points. Since $\forall i \in \{0, ..., m\}$, $\mathcal{M}_i \subset \mathcal{M}_{i+1}$, we have a filtration of $\mathcal{M}$:

$$\varnothing \subset \mathcal{M}_0 \subset \mathcal{M}_1 \subset \mathcal{M}_2 \subset ... \subset \mathcal{M}_m \subset \mathcal{M}$$

Morse theory tells us that $\mathcal{M}_{i+1}$ is homotopy equivalent (and thus they have the same homology groups) to the result of attaching a p-dimensional cell along the boundary of $\mathcal{M}_i$, where p is the index of the i-th critical point ([2]). Remember that the *index* of a non-degenerate critical point x is the number of negative eigenvalues of the Hessian in x.

Therefore, there are only two possibilities for how homology may change when we pass from $\mathcal{M}_i$ to $\mathcal{M}_{i+1}$:

(a) If the rank of H_p increases by one, we call the critical point p_i *positive*. In this case $\beta_p(\mathcal{M}_{i+1}) = \beta_p(\mathcal{M}_i) + 1$.

(b) If the rank of H_p decreases by one, we call the critical point p_i *negative*. In this case $\beta_p(\mathcal{M}_{i+1}) = \beta_p(\mathcal{M}_i) - 1$.

In order to track the born and death of classes, persistence pairs some of the positive critical points of index p (that represents the birth of a class) and some of the negative critical points of index p+1 (that represents the death of a class). But how exactly does persistence do this pairing?
Consider a class α. As explained in section 1.2 we can easily calculate its birth set $\mathcal{M}_i$ and death set $\mathcal{M}_i$ by using the maps between homology groups induced by the inclusions in the filtration. Once we now the birth and death sets of α we pair their corresponding critical points x_1 and y_1, and say that the *persistence* of the class is $f(y_1) - f(x_1)$.

If we want to store all of this information in an efficient way, in the sense of computability, instead of barcodes we can use the *persistence diagrams*, $Dgm_p(f)$. Each $Dgm_p(f)$ is a set (they are sets for Morse functions since only one class can be born or die at the same time, because critical points have all different values. In other filtrations, persistence diagrams are multisets) which includes the point $(f(x_1), f(y_1))$ whenever x_1 is a positive critical point of index p paired with the negative critical point y_1 of index p+1. All those points live in the half-space above the diagonal x = y and the persistence of each point can be graphically understood as the vertical distance between the point and the diagonal. Also, we include the *essential classes* of $\mathcal{K}$, those classes which do not die within the filtration. To include those, we add the pair $(f(x_1), \infty)$, where x_1 is the critical point corresponding to the birth set $\mathcal{M}_i$ of the class.

Note that barcodes and persistence diagrams hold the same information. There is actually a bijection between them, and even though we defined barcodes for Čech and Rips complexes, and persistence diagrams for Morse functions, both can be used in the other context mutatis mutandis.The difference between them is mainly their usage: the barcodes are a more didactic representation of persistence,

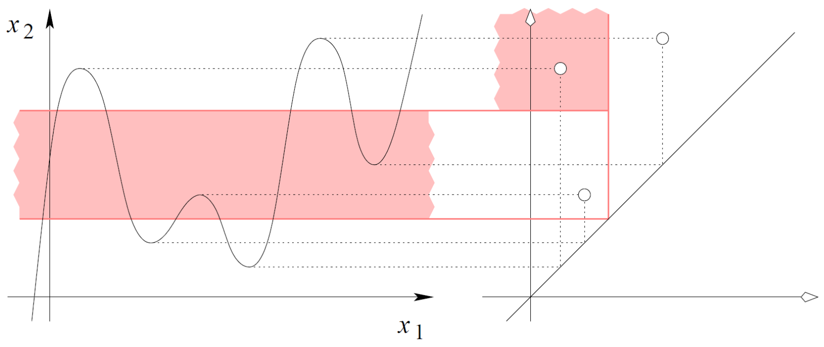

Figure 1.7: This image was taken from [5]. On the left, a single variable (Morse) function with 6 critical points. On the right, critical points are paired and shown in a persistence diagram.

while the persistence diagrams is becomes useful in the context of research and when there are too many classes to visualize them as barcodes.

1.6 Stability

The property that makes persistent homology a useful tool is its stability under perturbations of the topological space. That means that, for little changes in the space, little changes appear in its persistent homology. Let's first define a distance between persistence diagrams.

Definition 1.9. Let $\mathcal{M}$ be a topological space and $f, g : \mathcal{M} \to \mathbb{R}$ be two Morse functions. The *bottleneck distance* between the two diagrams $Dgm_p(f)$ and $Dgm_p(g)$ is defined as:

$$d_B(Dgm_p(f), Dgm_p(g)) = \underset{\eta}{inf} \ \underset{x}{sup} \ \|x - \eta(x)\|_\infty$$

where η is any bijection between the two persistence diagrams and $\|.\|_\infty$ is the ∞-norm defined as $\|x\|_\infty = max|x_i|$.

It is important to specify that the diagrams also includes copies of all points in the diagonal. This is necessary because the number of off- diagonal points in two different diagrams might not be the same and a bijection η between the two would not be possible.

The following theorem shows the first step to prove the stability of persistence, bounding the bottleneck distance of the diagrams with the distance between functions.

Theorem 1.10. *Let $\mathcal{M}$ be a topological space and $f, g : \mathcal{M} \to \mathbb{R}$ be two Morse functions. Then, for each dimension p, the bottleneck distance between the dimension p persistence diagrams is bounded from above by the distance between the functions,*

$$d_B(Dgm_p(f), Dgm_p(g)) \leq \|f - g\|_\infty$$

The infinity norm of a function is defined as $\|f\|_\infty = \sup_{x \in \mathcal{M}} |f(x)|$.

Proof. The proof of this proposition can be found in [3]. □

Let's now see some immediate consequences of persistence diagrams stability.

1.6.1 Homology inference

Let $\mathcal{M}_0$, $\mathcal{M}_1 \subset \mathbb{R}^d$ be two closed sets. For the purpose of this context, consider we want to study the topological features of $\mathcal{M}_0$ but it is much easier to calculate the persistent homology of $\mathcal{M}_1$. If $\mathcal{M}_0$ and $\mathcal{M}_1$ are "similar", how can we relate the features of those objects?

First, we will specify what we mean when we say "similar". For this purpose, let's define a distance between two sets, the *Hausdorff distance*.

Definition 1.11. Let $\mathcal{M}_0$ and $\mathcal{M}_1$ be two subsets of a metric space. We define the *Hausdorff distance* between $\mathcal{M}_0$ and $\mathcal{M}_1$ as

$$d_H(\mathcal{M}_0, \mathcal{M}_1) = inf\{\varepsilon \geq 0; \mathcal{M}_0 \subset \mathcal{M}_1^\varepsilon \text{ and } \mathcal{M}_1 \subset \mathcal{M}_0^\varepsilon\}$$

where $\mathcal{M}_i^\varepsilon = d_i^{-1}[0, \varepsilon]$ and $d_i : \mathbb{R}^d \to \mathbb{R}$ is the Euclidean distance function that maps each point to its Euclidean distance from the nearest point of $\mathcal{M}_i$, for $i \in \{0, 1\}$.

Ultimately, we define the *homological feature size* of $\mathcal{M}_0$ as the infimum of the positive homological critical value of d_0, and we denote it $hfs(\mathcal{M}_0)$.

The following proposition solves the initial problem:

Proposition 1.12. *Let $\mathcal{M}_0$, $\mathcal{M}_1 \subset \mathbb{R}^d$ be two closed sets.*
For any ε such that $d_h(\mathcal{M}_0, \mathcal{M}_1) < \varepsilon < hfs(\mathcal{M}_0)$ and a $\delta > 0$ sufficiently small,

$$rank\ H_p(\mathcal{M}_0^\delta) = rank\ im\ f_\varepsilon^{3\varepsilon}$$

where $f_\varepsilon^{3\varepsilon} : H_p(\mathcal{M}_1^\varepsilon) \to H_p(\mathcal{M}_1^{3\varepsilon})$ is the map induced by the inclusion $\mathcal{M}_1^\varepsilon \subset H_p(\mathcal{M}_1^{3\varepsilon})$.

Proof. The proof of this proposition can be found in [3] as a corollary of Theorem 1.10. □

For example, we could use this property if we want to study $\mathcal{M}_0$, a body bounded by a smooth surface. We can take $\mathcal{M}_1 \subset \mathcal{M}_0$ a finite point sample of the body, and deduce the rank of the persistent homology groups of $\mathcal{M}_0$ from the persistent homology of the filtration $\{\mathcal{M}_1^{\epsilon}\}_{\epsilon>0}$. Notice that $\{\mathcal{M}_1^{\epsilon}\}_{\epsilon>0}$ it is a filtration of Čech complexes of the collection of points $\mathcal{M}_1$.

1.6.2 Shape comparison

Consider we have $\mathcal{M}_0, \mathcal{M}_1 \subset \mathbb{R}^3$ and we know their Hausdorff distance is relatively small. How much can their topological features differ?

Proposition 1.13. *Let $\mathcal{M}_i \subset \mathbb{R}^3$ and $d_i : \mathbb{R}^3 \to \mathbb{R}$ the Euclidean distance function that maps each point to its Euclidean distance from the nearest point in $\mathcal{M}_i$, for $i \in \{0,1\}$. Then,*

$$\|d_0 - d_1\|_\infty \leq d_H(\mathcal{M}_0, \mathcal{M}_1)$$

Proof. Suppose that for given $\mathcal{M}_0, \mathcal{M}_1 \subset \mathbb{R}^3$, we have that $l := \|d_0 - d_1\|_\infty > d_H(\mathcal{M}_0, \mathcal{M}_1)$. Then, by definition of Hausdorff distance, there are two possibilities: $\mathcal{M}_0 \not\subseteq d_1^{-1}[0,l]$ or $\mathcal{M}_1 \not\subseteq d_0^{-1}[0,l]$.

We suppose the first case (The other one holds the same demonstration mutatis mutandis exchanging $\mathcal{M}_0$ with $\mathcal{M}_1$, and d_0 with d_1). This means that for some point in $\mathcal{M}_0$, there is no point of $\mathcal{M}_1$ within a distance of l. i.e. $\exists p \in \mathcal{M}_0$ such that $\forall x \in \mathcal{M}_1$, $d_E(p,x) > l$, where $d(,)$ is the Euclidean distance map in $\mathbb{R}^3$. Equivalently, $d_1(p) > l$, since we have defined $d_1()$ as the Euclidean distance function that maps each point to its Euclidean distance from the nearest point in $\mathcal{M}_1$. But we also know that $d_0(p) = 0$ since p is a point of $\mathcal{M}_0$. Finally, we get the following contradiction,

$$|d_0(p) - d_1(p)| = d_1(p) > l = \sup_{x \in \mathbb{R}^3} |d_0(x) - d_1(x)|$$

thus proving the proposition. □

If we use the last proposition in combination with Theorem 1.10, we get that the distance between the persistence diagrams of d_0 and d_1 is bounded from above by the Hausdorff distance of their corresponding sets $\mathcal{M}_0$ and $\mathcal{M}_1$.

$$d_B(Dgm_p(d0), Dgm_p(d1)) \leq \|d_0 - d_1\|_\infty \leq d_H(\mathcal{M}_0, \mathcal{M}_1)$$

Thus, a small Hausdorff distance implies similar topological features between the shapes.

Chapter 2

Adversary detection

The objective of this chapter is to show four methods of adversary detection in neural networks using persistent homology. They are presented in the paper [4][1]. In order to do so, we first need to have a brief introduction to what a neural network and its adversary examples are.

Before getting into it, i just want to remark that, since this is not a computer science report, we are not going to get into details for what a neural network is or the different types of construction we can make. we will just stick to the math involved in the problem. What follows is simply a short introduction to neural networks so the reader knows what this is all about.

2.1 Neural networks

A neural network (NN) is a function that, through a process of training, learns how to recognize patterns that distinguish classes of a concrete type of data and this way it is able to classify them. A common example of this is a neural network that, when given a n x n pixels image, is able to identify which digit from 0 to 9 it represents.

As its name says, a neural network is composed of *neurons*, and connections between them that create a network, called *edges*. The neurons receive inputs from other cells, process the information and store the output as a real number. Then, they send the output to connected neurons. The edges are weighted connections between them. They receive the output of the first cell, increase or decrease the value proportionally to their *weights*, and send the result to the other cell.

The neurons are aggregated into layers. This way, the information flows from the first layer (the *input layer*) to the second, from the second to the third,... succes-

[1]In the paper they are presented as three methods, but the last one can be thought as two different ones, and that is why I decided to structure this way.

sively, until the last layer (the *output layer*). The input layer receives the information from outside the neural network, and the last shows its prediction.

The neural networks are trained by processing examples, each of which has an input, that is given to the neural network, and the result. Once the neural network has processed the input and has given its prediction, we compare it with the real result. The difference is the error. The network then adjusts its weighted connections according to a learning rule and using this error value. Through a succession of attempts and readjustments, the NN's error keeps decreasing, and the training can be terminated once a certain criteria is met.

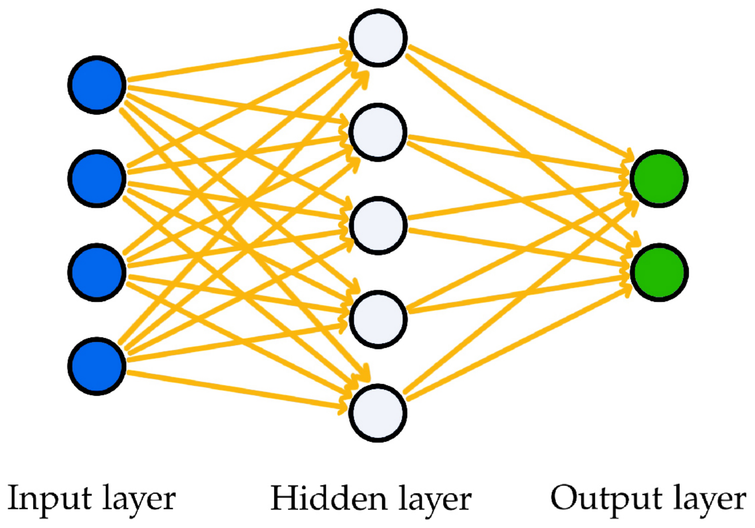

Figure 2.1: A graphical conceptualization of a simple 3-layers neural network. Blue neurons represent the input layer, white neurons form the hidden layer and green neurons represent the output layer. Neurons from adjacent layers are connected by edges (yellow arrows).

Now that a general idea of neural network has been given, we can show the notation that will be used for the rest of this chapter. Consider a neural network as a function $F : \mathbb{R}^n \to \mathbb{R}^m$ that sends each input $x \in \mathcal{U} \subset \mathbb{R}^n$, where $\mathcal{U}$ is the collection of examples we have available, to its class prediction $y \in \mathbb{R}^m$. The neural network used in this work computes the result using a softmax function, forcing each element y_i of y to the range $0 \leq y_i \leq 1$, and such that $\sum_{i=1}^{m} y_i = 1$. This way, the result $y_i \in y$ can be thought as the probability of an input x to belong to the class i. We define the *predicted class* of x as $C(x) = argmax_i\ F(x)_i$ and denote the *real class* of x as $C^*(x)$.

2.1.1 Adversary examples

Even though we can end up with a really "well trained" neural network, there are always some examples which get misclassified. In this work we focus on detecting a special type of these: the adversarial examples. Consider an input $x' \in \mathcal{U}$ which is close to another input $x \in \mathcal{U}$ according to some distance metric. When the input x is introduced in the neural network, it produces the correct class prediction $C^*(x)$. When we introduce x' in the NN, if the class prediction produced is $C(x') = t \neq C^*(x)$ and thus the input gets misclassified, we call x' a *targeted adversarial example* with target t. The goal of this work can then be resumed as:

For a given input $x \in \mathcal{U}$ with $C(x) = C^*(x) = t$ we want to minimize $\|x - (x + \delta)\|$, for some norm $\|.\|$, such that $C(x + \delta) = t_\delta \neq t$ with $x + \delta \in \mathcal{U}$.

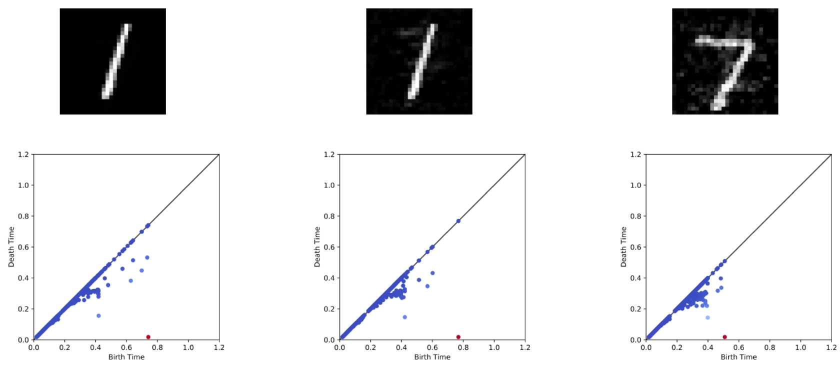

Figure 2.2: These images from [4] show the persistence diagrams of 3 different inputs (the 3 pixeled images on top). The left image gets correctly classified by the neural network as a 1. The middle one is an adversarial image generated from class 1 that is misclassified as a 7 by the neural network. The right one shows an adversarial image generated from class 7 that is misclassified as a 1. As we can see, the first and second persistence diagrams are very similar, despite the NN classifying them as different classes. On the other hand, the first and third images have very different persistence diagrams, but the NN makes the same classification for both.

2.1.2 Neural networks as graphs

"The results of this paper rely on the ability to represent a neural networks as computational graphs", as explained in [4]. One can already imagine that we will represent the NN's neurons as the vertices of the graph and the NN's edges as the edges of the graph. But, as we will see, things get a little more complicated.

Let's resume the main points our graph needs to fulfill:

- *Input depending*. The graph cannot be a static representation of the neural network, i.e. it cannot be defined only by the weights of the edges. We want it to represent the realized information flow after an input is added, since what we want to study is its response for different inputs.

- *No sign*. The graph should only represent the intensity (the absolute value) of the information that flows through the NN, not its sign. This is because, as we will see later, we are looking for which are the neurons and edges that have a stronger activation (again, depending on the input), no matter if they pass stimulant signals to the next layer (positive values) or suppressant signals (negative values).

- *Large-magnitude edges = proximity*. The graph neurons proximity needs to be higher when the NN's edge that connects them carries a large information activation. That is, high-weight NN's edges implies low distance between vertices, and thus, low-weight graph edges.

Those last 2 points are mostly motivated by our intuition on how information flow rate should be interpreted.

Before we continue with the graph representation of the neural network we need to specify that all this idea is well-founded because we work with convolutional, recurrent and fully-forward neural networks (see [4]). Also, we assume that the neural network has been trained such that the weight of the edges remains fixed when we apply more inputs.

Consider the neural network $F : \mathbb{R}^n \to \mathbb{R}^m$ that we defined in the last section. First, we define the the graph $\mathcal{G}$ as $G = (V, E, w)$ where V is the set of vertices (the neurons in the NN), E is the set of edges that connects them (the edges of the NN), and $w : E \to \mathbb{R}$ is the weight function that maps each edge to its correspondent weight in the neural network. Now that we have a base graph to work with, we will readjust it so it fulfills the 3 points previously explained.

First, consider an input $x \in \mathcal{U}$. When applied in the neural network it induces a new weight function $f : E \to \mathbb{R}$ based on the previous map w [2]. Depending on how the neural network is constructed, many edges may end up having zero weight, $f(e) = 0$. Those edges carries no information flow, and thus, can be neglected. That is why we define $E_x \subset E$ the set of all non-zero-weighted edges of E, $V_x \subset V$ [3] and $f' = f|_{E_x} : E_x \to \mathbb{R}$ the restriction of f over E_x. We then have a new graph $G_x = (V_x, E_x, f')$ which is input-depending.

Consider then the function $\phi = |f'|$ and the graph $G^*_x = (V_x, E_x, \phi)$. This way our new graph weights no longer carries a sign and we only need the graph to fulfill the last point.

This can be easily accomplished by defining m as the maximum edge weight in the graph, $m = max(\phi)$, and $\phi' = m - \phi$ as our new weight function. The final graph $G^{**}_x = (V_x, E_x, \phi')$ fulfill all the main 3 points and we can begin the persistent homology calculation.

2.2 Persistent homology calculation

As mentioned in the previous section, our goal is to show which are the substructures of the neural network which hold a stronger activation depending on the input. We do this using persistent homology.

First, we need a filtration of Rips complexes. This can be achieved by viewing G^{**}_x as a geometric realization of a simplicial complex. Remember that, as explained in section 1.2, the 1-skeleton fully determines the Rips complexes and thus the graph G^{**}_x retains all the informaion in order to construct our filtration.

Each Rips complex $\mathcal{R}_{ri}$ will be build on a set of points which represent the vertices of our graph. We then consider that the distance between each pair of points is given by $\phi'(e)$ where e is the edge connecting the corresponding vertices in the graph. If two vertices are not connected we will assume that their respective points in the complex are within a distance of ∞.
This way we can build the filtration:

$$\mathcal{R}_{r_0} \subset \mathcal{R}_{r_1} \subset ... \subset \mathcal{R}_{r_{p-1}} \subset \mathcal{R}_{r_p} \tag{2.1}$$

[2]In paper [4], not much is said about how the function f is constructed. The only thing it specifies is that $f = w \circ g$ where w is the previous weight function and $g : \mathbb{R}^n \to E$ depends on the network and the input. I've noticed that there is a notation problem, since the input set of g does not correspond with the input set of f. Also a reinterpretation of g as $g : E \to E$ would not make much sense in this context and this is why i decided to omit this part.

[3]In paper [4], there is no explanation about why we take a subset V_x of the original set of vertices V. I assume that the reason is that when we take the subset of edges E_x instead of E, some neurons might remain completely isolated from others and thus should not be considered since they have no implication in the NN for the input x.

Notice that for $r > m$, the Rips complexes in the filtration stabilize since all pair of points are within a distance of m (remember that $\phi' = m - \phi$) except those vertices that are not connected.

Before getting the calculation of persistence, we need to understand what feature we want to study and what they represent. First of all, we are only interested in H_0 and H_1, since the structure of neural networks implies a trivial homology H_p for $p > 1$. Also, as commented in [4], *"We found in our analysis that information from H1 is not especially meaningful for adversary detection"*. The paper explains that this is probably due to the fact that there are few 2-simplices[4], and thus almost all H_1 components are infinitely lived. In resume we will focus on studying H_0, tracking the most solid connected components.

As explained in section 1.3, we calculate the persistent homology H_0 of the filtration 2.1 and the corresponding persistence diagram which we will denote D. Remember that D is the set of birth-death pairs (b, d) of the classes in H_0. Also, for each class $\alpha \in H_0$ we consider the correspondent subgraph[5] and denote it G_x^α. We define a function $\tau : \{Class\ subgraphs\} \to D$ that maps each subgraph with its correspondent point in D. Now, we want to select only those subgraphs with higher lifetime. That is why we define a treshold $\lambda > 0$ and calculate the *persistent subgraph* $G_x^\lambda = \bigcup\{G_x^\lambda \mid G_x^\alpha = \tau^{-1}(b, d),\ b - d = l > \lambda\}$ which is an actual subgraph of G_x^{**}. We will use this new graph in order to compare the response of the neural network depending on the input.

The intuitive idea is that the subgraphs corresponding to classes with longer lifetimes $l = d - b$ are those most strongly associated to classification decisions, since they hold the higher information flow given an input x. And that is why we look for differences in those subgraphs in order to detect adversary examples.

2.3 Adversary detection

In this section we finally present the four methods of adversary detection using the persistent subgraph that we presented in the previous section.

[4]The general structure of neural networks implies that 2 neurons in the same layer are not connected. This way there are no three 1-simplices forming a triangle and thus no 2-simplices. The paper says that "there are few" probably because in some special NN's like Feedback ANN, some connections do not follow the general structure and can connect different layers that are not consecutive.

[5]Paper [4] does not specify what the correspondent graph of a class is considered to be. I assume it refers to the set of points and edges that started defining the class.

First, we need to remark that usually neural networks contain million of neurons and edges. This makes the calculus of persistent homology (and thus the persistent subgraph) too expensive computationally speaking. This is why we consider another threshold $\rho \in [0, 1]$ which function is to limit the edges we consider for the computation of the persistent graph to only the $100\rho\%$ with largest weight.

In the context of the paper, they considered $\rho = 0.99$, since as they explain, most of the relevant information is retained in the top 1% of edges. The methods are prapared with a selected subset $\mathcal{U}_{train}$ of $\mathcal{U}$, and then tested with another subset of $\mathcal{U}$ which they denote $\mathcal{U}_{test}$.

2.3.1 Maximum node matching

As explained in [4], *"In this detection method, we look to detect discrepancies between the input-induced topological signature and the expected topological signature of the predicted class"*. Specifically, the objective is to detect which substructures (subgraphs) of the neural network has a higher activation when a particular class type of input is introduced. This way, if a given input provides an activation of the NN similar to that of the class we studied but the NN misclassifies it, we can expect it to be an adversary example.

Getting into details, we fix the parameters ρ and λ and compute the persistent subgraph $^{\lambda}_{x}$ for each $x \in \mathcal{U}_{train}$, as explained in the previous section. Then, for each class $y_i \in y$, we consider the set of vertices that appear in the persistent subgraphs corresponding to an input classified as y_i by the NN, $\mathcal{V}_i = \{v \in V_x^\lambda \mid G_x^\lambda = (V_x^\lambda, E_x^\lambda, \phi'),\ x \in \mathcal{U}_{train},\ C(x) = i\}$. Notice that $\mathcal{V}_i \subset V$. But we also want to keep track of the number of appearance that each vertex has in the studied graphs. The reason is that a vertex with a higher appearance than others, has more chances of being representative of the class. That is why we define a function $r_i : \mathcal{V}_i \to \mathbb{Z}$, which pairs each vertex with its appearance in the graphs.

Resuming, we have a set $\mathcal{V}_i$ and a appearance function r_i for each class $y_i \in y$. The hypothesis this all method is based on is that these vertex sets should represent the most persistent semantic subgraphs used in the classification in each class. We then want to use the information recorded with the inputs of $\mathcal{U}_{train}$ to create a function that assigns to each input of $\mathcal{U}_{test}$ a score proportional to its similarity to a given class i. With this purpose, we define a new appearance function that extends the domain of r_i to all V:

$$m_i(v) = \begin{cases} r_i(v) & \text{if } v \in \mathcal{V}_i \\ 0 & \text{otherwise} \end{cases}$$

Then, for each class $y_i \in y$, we define its similarity score as:

$$s_i(G_x^\lambda) = \frac{\sum_{v \in V_x^\lambda} m_i(v)}{|\mathcal{V}_i|}$$

We denote i^* the class that shows higher similarity score with the tested graph, $i^* = argmax_i(s_i(G_x^\lambda))$. For a given $x \in \mathcal{U}_{test}$, if the NN's predicted class $i' = argmax_i(y)$ does not coincide with $i*$, we expect x to be an adversary example. The results of this process presented in the paper [4] are shown in the table of Figure 2.5.

2.3.2 Average node matching

"In defining the maximum node matching detection algorithm, we noticed that adversarial inputs generally have higher similarity scores to all classes than non-adversarial inputs"[6], they explain in paper [4].

This new method we are about to present in this section is motivated by this observation. First we take a new subset $\mathcal{U}_{val} \subset \mathcal{U}_{train}$[7]. Then, we consider that $y = (y_i)_{i=0}^{m-1}$. We calculate the mean number of matches (instead of the maximum edge class as in the previous method) as follows:

$$s_{avg}(G_x^\lambda) = \frac{\sum_{i=0}^{m} s_i(G_x^\lambda)}{m}$$

We compute $s_{avg}(G_x^\lambda)$ for each $x \in \mathcal{U}_{val}$ and we use the results to compute the mean and the standard deviation over all observations:

$$\mu_{val}^m = \frac{\sum_{x \in \mathcal{U}_{val}} s_{avg}(G_x^\lambda)}{|\mathcal{U}_{val}|}$$

$$\sigma_{val}^m = \sqrt{\frac{\sum_{x \in \mathcal{U}_{val}} (s_{avg}(G_x^\lambda) - \mu_{val}^m)^2}{|\mathcal{U}_{val}| - 1}}$$

If a given input $x \in \mathcal{U}_{test}$ shows a mean number of matches such that $s_{avg}(G_x^\lambda) > \mu_{val}^m + \sigma_{val}^m$ we can consider it has a higher general similarity score with classes then the average input and we can and tag it as an adversarial example. The results of

[6]The reason of this might be a consequence of the fact that the similarity score of the Maximum node matching method is not "punished" when the graph G_x^λ has vertices that do not usually show for inputs of the class i (i.e. $m_i(v) = 0$). Thus, a persistence graph containing all vertices of V would get the maximum similarity score for all classes.

[7]In paper [4], it is not specified why we use this new subset instead of using all $\mathcal{U}_{train}$. I suspect this decision comes after the necessity of taking a specific sample which makes the following statistic calculus more solid, but no further explanation is given by the paper.

this process presented in the paper [4] are shown in the table of Figure 2.6.

The last two methods we presented study the similarity between node sets of graphs. The next two, on the other hand, are focused on comparing the edge sets.

2.3.3 Edge counting

This next method is motivated by the property shown in Figure **??**. We observe that the persistent subgraphs of adversarial inputs contain many more edges compared to non-adversarial subgraphs. This only happens for values of λ around 0.1 as explained in paper [4]. We use this propery defining the following method.

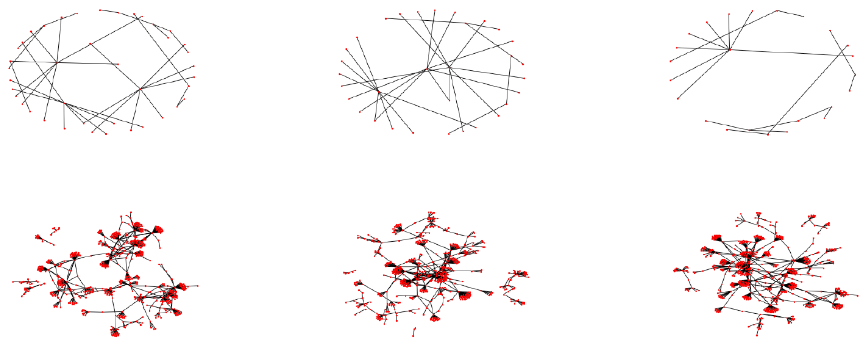

Figure 2.3: These images from [4] show force-directed subgraphs from all components with lifetime > 0.1 for unperturbed MNIST images (On top) and adversarial images (On the bottom) with same target class.

First, we define $\mathcal{E}_{train} = \{|E_x| \mid E_x \text{ is the edge set of the graph } G_x^\lambda,\ x \in \mathcal{U}_{train}\}$. We then calculate the median of this set $m_{train} = median(\mathcal{E}_{train})$ and the π^{th} percentile of edge counts in $\mathcal{E}_{train}$, and denote it p_{train}. This way, if a given input $x \in \mathcal{U}_{test}$ results having a persistent graph edge count such that $|E_x| > m_{train} + p_{train}$, we tag it as an adversarial example. The results of this process presented in the paper [4] are shown in the table of Figure 2.7.

2.3.4 Average edge weight

Another useful property of the edge sets in persistent graphs that we can use is the fact that *"the average edge weight amongst persistent subgraphs induced by adversarial images is both lower and less variable than those induced by non-adversarial images"* as explained in [4] and shown in Figure 2.4. This property is used by the following method in a way similar to the Average node matching method.

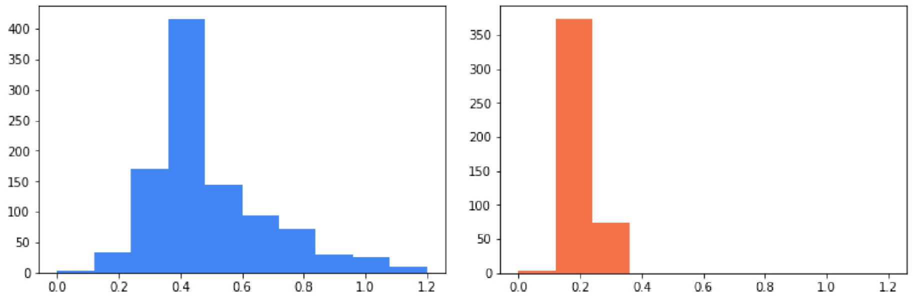

Figure 2.4: This graphics from [4] show the distribution of average edge weight for non-adversarial images (On the left) and for adversarial images (On the right).

First we calculate the average edge weight of the persistent subgraphs of each $x \in \mathcal{U}_t rain$ as follows:

$$r_{avg}(G_x^\lambda) = \frac{\sum_{e_x^\lambda} \phi'(e)}{|E_x|}$$

Then, we compute the mean and the standard deviation over all observations:

$$\mu_{train}^{\phi'} = \frac{\sum_{x \in \mathcal{U}_{train}} r_{avg}(G_x^\lambda)}{|\mathcal{U}_{train}|}$$

$$\sigma_{train}^{\phi'} = \sqrt{\frac{\sum_{x \in \mathcal{U}_{train}} (r_{avg}(G_x^\lambda) - \mu_{train}^{\phi'})^2}{|\mathcal{U}_{train}| - 1}}$$

If a given input $x \in \mathcal{U}_{test}$ shows an average edge weight such that $r_{avg}(G_x^\lambda) > \mu_{train}^{\phi'} + \sigma_{train}^{\phi'}$ we tag it as an adversarial example. The results of this process presented in the paper [4] are shown in the table of Figure 2.8.

2.4 Results

What follows are the results of the testing for the four adversary example detection presented in paper [4]. Each algorithm has been tested on a sample of 900 images, 450 of which were adversarial examples, and the rest were unaltered images. ρ was set on $\rho = 0.99$ and tests were made with many different values for λ. A last parameter κ was used, with values $\kappa = 0$ and $\kappa = 20$. κ controls the confidence with which the neural network misclassifies the generated adversarial images.

As can be seen in the following tables, this methods tend to be more successful with $\kappa = 20$ (i.e. with adversarial inputs which the NN misclassifies with high confidence), and in particular with $\lambda = 0.1$.

Now, if we focus on the best results for each value of κ (in bold in Figures 2.5, 2.6, 2.7 and 2.8), we see that the F1 scores are really close to 1, showing that those are promising methods for adversary detection. In particular, the average edge counting method shows the best results, between those 4 methods, in both confidence levels $\kappa = 0$ and $\kappa = 20$.

κ	λ	Accuracy	False Positives	False Negatives	F_1-score
0	0.1	0.683	235	50	0.737
0	0.05	0.786	169	23	0.816
0	0.01	0.858	109	**19**	0.871
0	0.001	**0.864**	**102**	20	**0.876**
20	0.1	**0.925**	**61**	**6**	**0.93**
20	0.001	0.854	102	29	0.865

Figure 2.5: Results of the maximum node matching method in [4]. The top results for both κ values are in bold.

κ	λ	Accuracy	False Positives	False Negatives	F_1-score
0	0.1	**0.885**	**45**	**58**	**0.884**
0	0.05	0.478	52	418	0.12
0	0.01	0.476	57	414	0.133
0	0.001	0.481	54	413	0.137
20	0.1	**0.938**	**45**	**11**	**0.94**
20	0.001	0.464	54	428	0.084

Figure 2.6: Results of the average node matching method in [4]. The top results for both κ values are in bold.

κ	λ	π	Accuracy	False Positives	False Negatives	F_1-score
0	0.1	0.9	**0.969**	**18**	**10**	**0.969**
0	0.05	0.9	0.816	7	158	0.78
0	0.01	0.9	0.786	6	168	0.733
20	0.1	0.95	**0.984**	**14**	**0**	**0.985**

Figure 2.7: Results of the average edge counting method in [4]. The top results for both κ values are in bold.

κ	λ	Accuracy	False Positives	False Negatives	F_1-score
0	0.1	0.914	**30**	47	0.913
0	0.05	**0.931**	33	**29**	**0.931**
0	0.01	0.857	55	74	0.854
0	0.001	0.788	122	69	0.774
20	0.1	**0.95**	**30**	**15**	**0.951**
20	0.001	0.865	69	52	0.868

Figure 2.8: Results of the average edge weighting method in [4]. The top results for both κ values are in bold.

Conclusions

During the process of writing this work I've learned many things. First, it helped me understand homology in a more graphical way. Studying all these different complexes and shapes helped me relate my visual intuition with the algebraic theory that I had already learned in class.

Secondly, I learned to be more accurate in my explanations since for the first time in my life I had to use my mathematical knowledge, not with the objective of proving that I know something, but in order to make sure that the reader understands what I try to show her/him.

Finally, and for me the most important thing, I worked in my favourite field in mathematics, topology, and discovered many applications it has nowadays.

Bibliography

[1] Hatcher, Allen. *Algebraic Topology*. Cambridge University Press, 2002.

[2] J. MILNOR. *Morse Theory*. Princeton Univ. Press, New Jersey, 1963.

[3] D. COHEN-STEINER, H. EDELSBRUNNER AND J. HARER. *Stability of persistence diagrams. Discrete Comput. Geom.* 37 (2007), 103-120.

[4] THOMAS GEBHART and PAUL SCHRATER. *Adversary Detection in Neural Networks via Persistent Homology*, 2017.

[5] Edelsbrunner, Herbert, and John Harer. *Persistent homology - a survey*. Contemporary mathematics 453 (2008).

[6] Li, M, Duncan K, Topp CN, Chitwood DH (2017) *Persistent homology and the branching topologies of plants*. Am J Bot 104(3): 349 - 353.

[7] Github of *Fritz Pere Nobbe*. https://github.com/fritzpere/Gudhi_examples.

[8] Ghrist, Robert. *"Barcodes: the persistent topology of data."* Bulletin of the American Mathematical Society 45.1 (2008)