An Introduction to Topological Data Analysis for Data Scientists

Martin Berger

Contents

The following notes give a brief introduction to the modern field of Topological Data Analysis (TDA). Since TDA admits a nice geometric explanation, our focus lies hereby on a conceptual understanding by observing several exmaples, instead of a mathematical rigorous examination. Most of the examples down below are using the R package TDA, see (Fasy et al. 2014).

1 Topology

We start by giving a brief idea of (algebraic) Topology. Topologists try to classify topological spaces (think of a topological space as some geometric object, or simply the real line $\mathbb{R}$ or an

interval) which are preserved under continuous transformation, i. e. properties which are maintained after applying a continuous function to the geometric object. These continuous operations include deformations such as stretching, twisting, scaling and bending but they do not allow tearing or gluing. Intuitively spoken, for a topologist the world consists of objects which are made of rubber and if we are able to turn an object into another by means of continuous transformations, these objects are considered the same. For example, in this sense a sphere S is the same as a hollow cube C, because if the material is rubber, we can smoothly turn the cube into a sphere and vice versa. In this situation mathematicians write $S \cong C$, where the Tilde emphasizes that we mean equality in a certain sense. More particularly, what topologists mainly study are topological descriptions like the number of components (0-dimensional hole), the number of holes (1-dimensional hole), the number of voids (2-dimensional hole), and their higher dimensional equivalents. Clearly these numbers are preserved after deforming them continuously. If you for example shrink or blow up a donut, it will still have a hole in the middle. The hole will only vanish if we cut through it. This also explains why, if you google the phrase "topology math", you typically will find a picture where it is illustrated that a mug is the same as a donut. Both objects consist of one component and have one hole, thus they are considered the same. If you visit the wikipedia page https://en.wikipedia.org/wiki/Topology you will find a nice animation which turns a cow into a sphere with interior. Again the reasoning for this animation is, that they are considered the same. Both have one component and no holes or voids (naively spoken). However, while the above mentioned statements are geometrically intuitive and in some sense funny, it should be mentioned, that the underlying mathematical theory is very abstract and highly non trivial. Already a formal way of defining what a "hole" is, is quite sophisticated.

Figure 1: A donut is topologically the same as a mug.

2 Topological Data Analysis

2.1 Data has shape, and shape has meaning.[1]

The aim of topological data analysis is to identify the geometric structure within some finite statistical data points set D. Hence, we assume that our data points are randomly drawn

[1] Quote by Gunnar Carlsson

from a geometric object and by estimating the number of topological features we try to figure out, how this object looks like. Approaching datasets by means of algebraic topology is a quite novel approach and due to many successes TDA has achieved, its methods have gained more and more poularity over the past decade. For instance in medicine type 2 diabetes subgroups were identified using TDA (Li et al. 2015) as well as subgroups of breast cancers with a unique mutational profile (Nicolau, Levine, and Carlsson 2011). Another reason for its gaining usage is increasing computer performance. TDA is computationally expensive and thus often not applicable in practice. However, a lot of research has been conducted to improve existing and invent new algorithms, which allow applications with big data sets, i. e. high resolution images, which we will discuss in section 2.4.

2.2 From Point Clouds to Geometry

So we know our goal is to analyze the geometry of a given dataset. However, if we are handed a set of points D, per se there is not much topological structure to observe. Assume $\#D = 10$, then we know that D consists of 10 components. But obviously we can't talk about holes or voids. We therefore need to find a way in order to create a more sophisticated object, which admits more interesting features. The main tool we will use for this purpose is a so called simplex. Given a set of points $(p_0, \dots, p_m)$, which we assume to be affine independent[2], we call its convex hull

$$s = [p_0, \dots, p_m] \coloneqq \left\{ \sum_{i=1}^{m} \lambda_i p_i \mid \sum_{i=1}^{m} \lambda_i = 1 \, , \lambda_i \geq 0 \right\}$$

the m-simplex with vertices $\mathrm{Vert}(s) = \{p_0, \dots, p_m\}$. Often we will omit the dimension and simply write simplex. Given a simplex s a subsimplex is a simplex whose vertices form a subset of $\mathrm{Vert}(s)$. A proper subsimplex is often called a face. For instance a 0-simplex is simply a point, a 1-simplex is a line between two points and its faces are the two endpoints. A 2-simplex is a triangle (with interior!) and its faces are its vertices and edges. The assumption of being affine idependent is necessary, as otherwise in case of a 2-simplex, the three points would be colinear and therefore would not form a triangle. A set consisting of simplices such that also every subsimplex lies in the set, is called a (abstract) simplicial complex.

Figure 2: Example of a 1-simplex.

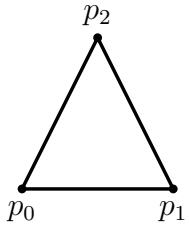


Figure 3: Example of a 2-simplex.

Our goal is now to create a family of simplices by using the given data set and analyze its topology. A very intuitive way is by using a metric d. Given a point $x \in \mathbb{R}^n$ we define the

[2] i.e. $p_1 - p_0, \dots, p_m - p_0$ are linearly independent.

(closed) ball with radius $r > 0$ around x as the set $B_r(x) = \{y \in \mathbb{R}^n \mid d(x,y) \leq r\}$. We are now able to formulate the following definitions.

Definition 2.1 (Cech and Vietoris - Rips Complex)**.** Let $r > 0$ and D a finite set.

i) We define the Čech Complex $\mathcal{C}^r$ as the set of simplices $[p_0, \ldots, p_m]$ and all their subsimplices, such that $p_0, \ldots, p_m \in D$ and it holds $B_{r/2}(p_0) \cap \ldots \cap B_{r/2}(p_m) \neq \emptyset$.

ii) We define the Vietoris-Rips Complex $\mathcal{R}^r$ as the set of simplices $[p_0, \ldots, p_m]$ and all their subsimplices, such that $p_0, \ldots, p_m \in D$ and $B_{r/2}(p_i) \cap B_{r/2}(p_j) \neq \emptyset$ for $i \neq j$.

In other words, in the case of a Čech Complex the simplices consist of vertices whose closed $r/2$-balls have a point of common intersection while in the Vietoris-Rips case the vertices lie pairwise within distance r. This subtle difference is illustrated in the following example. Assume d denotes the standard euclidean distance, i. e. for $x, y \in \mathbb{R}^n$ it is

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\,.$$

In Figure 4 we see a point cloud in the plane (top) as well as the corresponding Čech Complex (middle) and Vietoris-Rips Complex (bottom) for to some $r > 0$. In both Complexes we see that the rightmost point is not connected to any other point (i. e. there is no edge drawn between another point) since the distance to every other datapoint is larger than r, thus the closed balls with radius $r/2$ do not intersect. The difference between the two definitions can be seen, when we observe the leftmost triangle. The three vertices are pairwise within distance r and therefore the generated triangle is added to the Vietoris-Rips complex (illustrated by filling it with color). However, if you look closely, the three balls around the vertices do not share a common intersection point, thus the triangle is not an element of the Čech Complex. Whichever method you prefer, we see that for fixed r we are able to construct a set of simplices which admits some sort of geometry. For example we also see in Figure 4 that both complexes consist of 2 components, the big connected one and the single rightmost point, as well as that the Čech complex has two holes, the white triangles, whereas in the Vietoris-Rips complex no holes appear.

Checking for holes and components can be done algorithmically by using the so called Smith-Normalform but we won't go into detail about this procedure. However, the geometry we observe greatly depends on the chosen radius r. The idea is now to look at an increasing sequence of radii $0 \leq r_1 \leq r_2 \leq \ldots \leq r_N$ and to calculate a complex for every r_i in the sequence. Since we always included every subsimplex in the definitions of the above complexes, this gives rise to the following two increasing chains of sets

$$\mathcal{C}^0 \subseteq \mathcal{C}^{r_1} \subseteq \ldots \subseteq \mathcal{C}^{r_N}$$

and

$$\mathcal{R}^0 \subseteq \mathcal{R}^{r_1} \subseteq \ldots \subseteq \mathcal{R}^{r_N}\,.$$

This yields the the notion of a filtration.

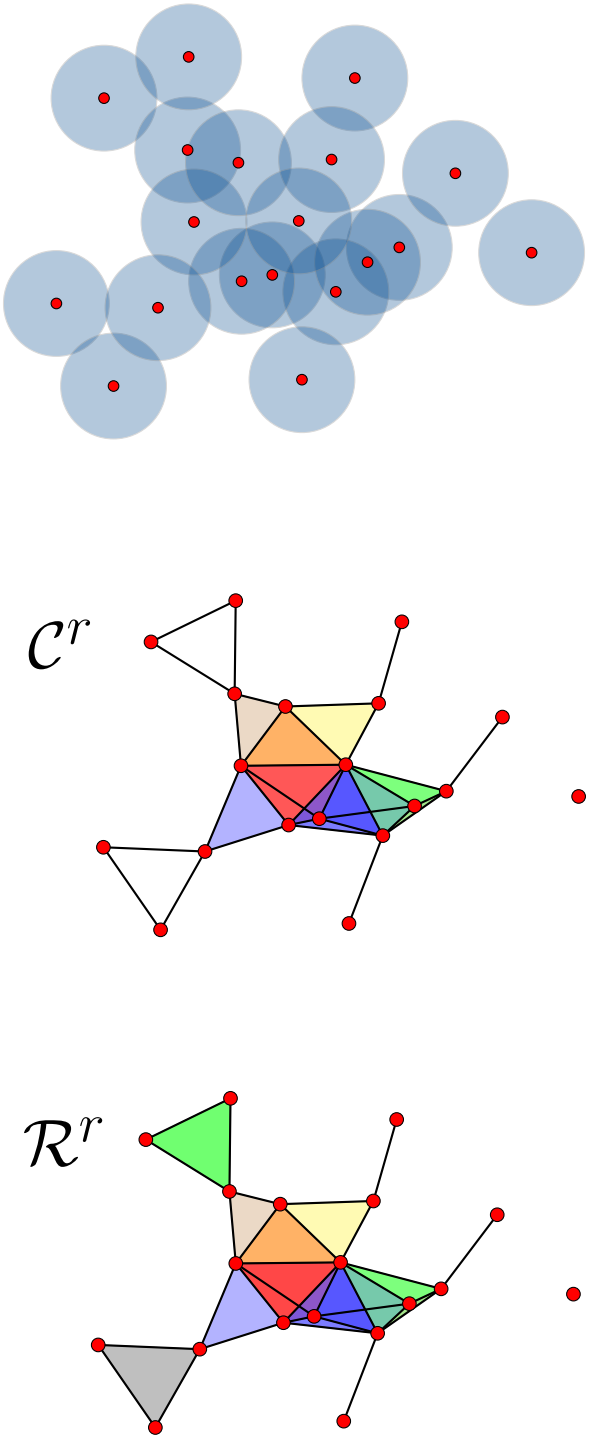


Figure 4: A point cloud in $\mathbb{R}^2$ where every point is surrounded by an r neighbourhood (top) and its corresponding Čech Complex (middle) and Vietoris-Rips Complex (bottom) for a fixed $r > 0$.

Definition 2.2 (Filtration of Complexes)**.** Given an (abstract) simplicial complex $\mathcal{K}$, a filtration is a totally ordered set of complexes $\mathcal{K}^i \subseteq \mathcal{K}$, for $i \in \mathbb{N}$, such that $i \leq j$ implies $\mathcal{K}^i \subseteq \mathcal{K}^j$.

We can generalize the above the constructions in the following way.

Definition 2.3 (Sublevel and Superlevel Filtration)**.** Let X denote a topological space and $f\colon X \to \mathbb{R}$ a continuous function. We define its sublevel set

$$X_t \coloneqq f^{-1}\Big((-\infty, t]\Big) = \{x \in X \mid f(x) \leq t\}\,.$$

Given an increasing number of real numbers $t_1 \leq \ldots \leq t_N$, f induces an increasing chain of topological spaces

$$X_{t_1} \subseteq \ldots \subseteq X_{t_N},$$

which we call the sublevel filtration generated by f. Analogously we define the superlevel filtration generated by f for a decreasing sequence $t_1 \geq t_2 \geq \ldots \geq t_N$.

Now given a filtration $X_{t_0} \subseteq \ldots \subseteq X_{t_N}$ we can proceed in the following way. For each t_i we calculate the number of topological features and measure how long a feature persists. For example a data point might form a component at t_0. By increasing the parameter t, two points might get connected by an edge when increasing the radius to t_1, and thus, the two components merge into one component. We therefore say this component has a lifespan or persistence $t_1 - t_0$. Analogously three edges can form a cycle, and thus forming a hole, at level t_2. Eventually this hole could be closed for a larger t_2 by filling a triangle into the cycle. Therefore we say, this hole, has a persistence of $t_3 - t_2$. Clearly, the greater the persistence of a feature is, the more likely our data is drawn from an object admiting this feature. Geometrical features with a small persistence however can be considered as noise. Clearly the outcome of our analysis depends completely on the filtration. Therefore, investing time into finding the right filtration, might give completely new insights to the data structure.

Again, this can be done algorithmically, how will not be discussed here. But it should be mentioned that the standard algorithm has cubic runtime in the number of simplices appearing. Since we need $n+1$ points to generate an n-simplex we obtain $\binom{\#D}{n+1}$ as an upper bound of the number of possible n-simplices. And since for a finite set D the maximal dimension of a simplex is $\#D - 1$ we get

$$\sum_{n=0}^{\#D-1} \binom{\#D}{n+1}$$

as an upper bound of the number of all possible occuring simplices.

2.3 Persistence Diagrams

Having calculated the topological features for a given filtration, our next goal is to illustrate the topology and transfer the measurements in a way, which allows us to analyze them statistically. One widely used way to do so is by using persistence diagrams. Recall that a multiset is a set, which allows multiple instances for each of its elements.

Definition 2.4 (Persistence Diagram)**.** A persistence diagram is a countable multiset of points in $\mathbb{R}^2$ together with the diagonal $\{(x, x) \mid x \in \mathbb{R}\}$ where each point on the diagonal has multiplicity infinity. The diagram consisting only of the diagonal is called the empty diagram.

As we see the definition of a persistence diagram is very general and counting each point on the diagonal with infinite multiplicity has technical reasons. Let's take a look at an example which should clearify its application.

2.3.1 Example: Sampling Points from a Double Torus

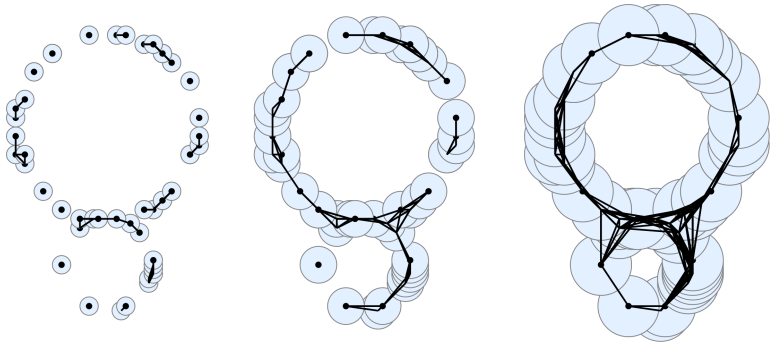


Figure 5: Three Vietoris-Rips complexes created from a point cloud uniformly drawn from two circles, only 1-simplices are drawn due to visibility.

We randomly sampled points from a double Torus and created a filtration consisting of Vietoris-Rips complexes. Figure 5 illustrates three complexes of the generated filtration. In the first complex (left) a very small radius is used, thus most of the points are not connected. Thus at this part of the filtration a lot of components will be measured. Increasing the radius further, we see in the second complex (middle), that now most of the sample points are connected by edges, leaving four components left. However the edges still have not enclosed the two holes we want to identify. But we see, that the upper big hole will be identified before the small one, since the edges around the smaller hole have a larger gap, than the ones above. Enlarging the radius even more (right), both holes are now captured by the edges. Clearly, we additionally see that while the big hole is identified earlier, it will vanish after the small hole, since we will need balls of larger radius in order to cover the upper gap.

Figure 6 shows the corresponding persistence diagram. Blue points represent components while red points represent holes. Futhermore the x-coordinate of a point is the radius for which the topological feature is "born" and the y-coordinate tells us at which radius the feature vanishes or "dies". Thus, a persistence diagram is often called a birth-death diagram. Obviously, since a point can only vanish after it is born, every point lies above the diagonal. As we can see, all components occur already at radius zero, which is clear since every point forms a component at this level. However increasing the radii slightly, we see that the components merge quickly, since points get connected by edges. This is perfectly illustrates by the blue points. Taking a look at the red points, we clearly recognize the two holes. Additionally their coordinates describe what we expected in Figure 5. The big hole will be born earlier but will vanish later, therefore it is represented by the red point with larger

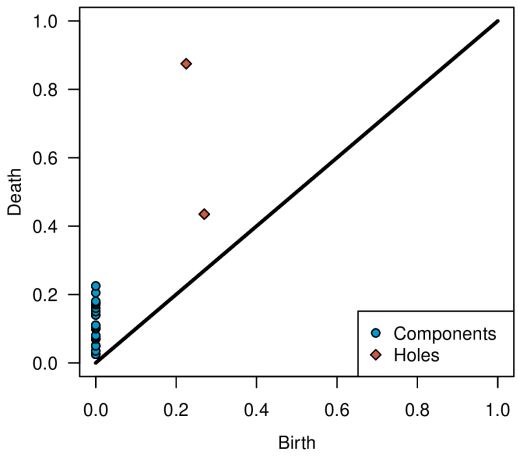


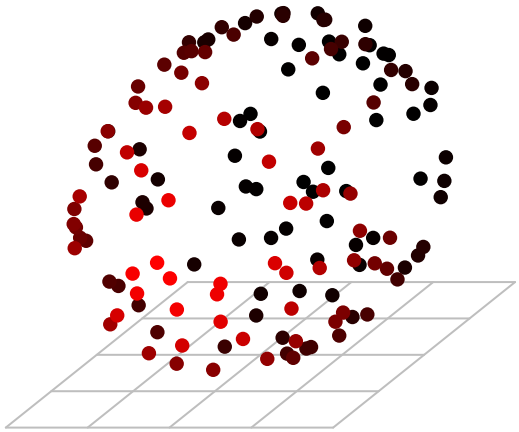
Figure 6: The persistence diagram corresponding to the example illustrated in Figure 5.

y-coordinate, while the smaller one, arises later, but vanishes earlier and therefore corresponds to the red point closer to the diagonal. Additionally, note that the persistence of a feature is given by the difference of its y- and x-coordinate. So in the above example, we see that a persistence diagram seems to be a nice way of illustrating the hidden geometry. However, keep in mind, that **each point in the diagram can represent multiple features**. In more complex situations several holes can be born and vanished at the same parameter level. Thus, the plot of the diagram alone, won't tell us necessarily enough information. We will discuss ways in order to do soon. Addtionally we see that the persistence of a hole depends highly on its size when we are using a Vietoris-Rips complex.

2.3.2 `R` Example: Persistence Diagram of a Sphere

It's time for our first example in `R`. The following code creates a point cloud uniformly drawn from a sphere and creates the corresponding persistence diagram using the package `TDA`.

```
library(TDA)
library(scatterplot3d)
set.seed(1)

# Creating the point cloud
n = 150
sphere = sphereUnif(n, d = 2, r = 1)

scatterplot3d(sphere,pch = 16, grid = TRUE, box = FALSE,
              highlight.3d = TRUE, asp = 1,pty="s", axis = FALSE)
```



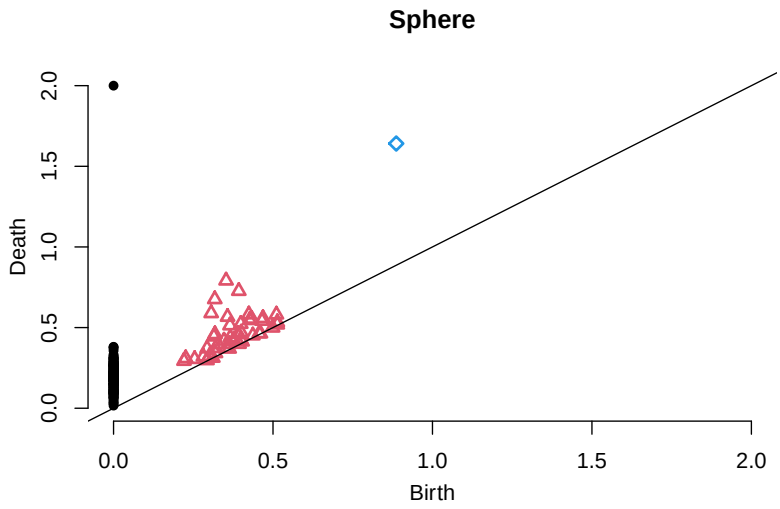
```
# The command ripsDiag creates a Vietoris-Rips Complex from a
# given point cloud. As metric, we use the euclidean distance.
# One can use an arbitrary distance function, by providing
# a distance matrix as input and by setting dist = "arbitrary"
sphereDiag = ripsDiag(sphere, maxdimension = 2, maxscale = 2,
                      dist = 'euclidean')

# The output is a list which shows us the dimension of a
# feature (0 for componens, 1 for holes, 2 for voids) and
# when the feature was born and when it dies
head(sphereDiag[[1]])
```

```
##      dimension Birth     Death
## [1,]         0     0 2.0000000
## [2,]         0     0 0.3806680
## [3,]         0     0 0.3789236
## [4,]         0     0 0.3712936
## [5,]         0     0 0.3570147
## [6,]         0     0 0.3328951
```

```
summary.diagram(sphereDiag$diagram)
```

```
## Call:
## ripsDiag(X = sphere, maxdimension = 2, maxscale = 2, dist = "euclidean")
##
## Number of features:
## [1] 194
##
## Max dimension:
## [1] 2
##
## Scale:
## [1] 0 2
```

```
# We can plot the persistence diagram with the TDA command
# plot.diagram
plot.diagram(sphereDiag$diagram, main = 'Sphere')
```



As we can see in the diagram, the Vietoris-Rips filtration clearly captures the void of the sphere (blue rectangle) and one persistent component. But we still obtain a lot of features around the diagonal. Due to their low persistence, they might be considered as noise.

2.3.3 Interpreting a Sublevel Filtration

While the above eample uses the geometric very intutive notion of a Vietoris-Rips complex, we now want to discuss a simple example, which illustrates the applications of a more abstract filtration. Let $f\colon X \to \mathbb{R}$ denote some function and let's consider the sublevel filtration. We can interpret the filtration in the following way. Since $X_t = \{x \in X \mid f(x) \leq t\}$, we can assume we are sliding a straight horizontal line from bottom to top along the y-axis and we subsequently add points x to X_t when $f(x)$ lies below the line at a height of t.

In Figure 7 we see on the left hand side such a function f and on the right hand side the corresponding persistence diagram, when using a sublevel filtration. The dashed lines represent the horzintal line we slide over the graph. Note, whenever the vertical line touches a local minima a new component is born and when we slide over a local minima, two components merge. If we would have used a superlevel filtration, the components would have risen at maxima and vanish at minima. These observations are also stable when noise is added. In the `R` code down below we construct a persistence diagram of the same polynomial, but this time we added noise. Clearly we still recognize the same three components as in the case without noise, while all the new components have a very short persistence and are therefore also considered as noise. Due to this noise robustness, TDA is also popular in time series analysis, see (Ravishanker and Chen 2019) or (Gholizadeh and Zadrozny 2018) for a survey on some applications.

```
xlim = c(0, 4.92)
by = 0.01
```

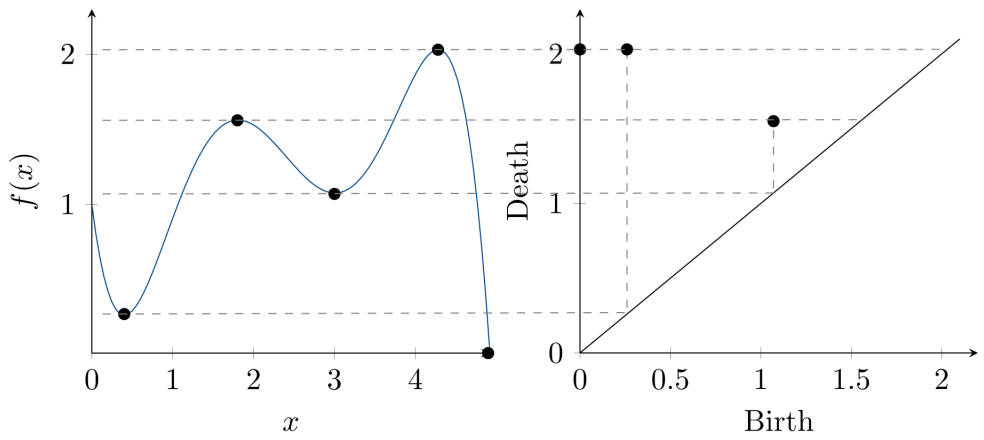
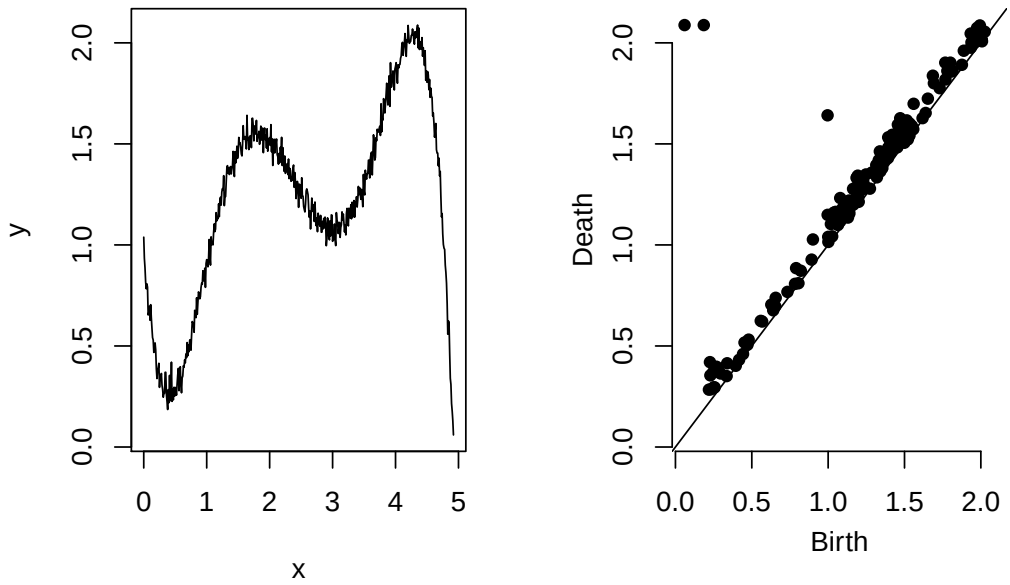


Figure 7: Left: The graph of a function f. Right: The corresponding persistence diagram for components of the sublevel filtration generated by f.

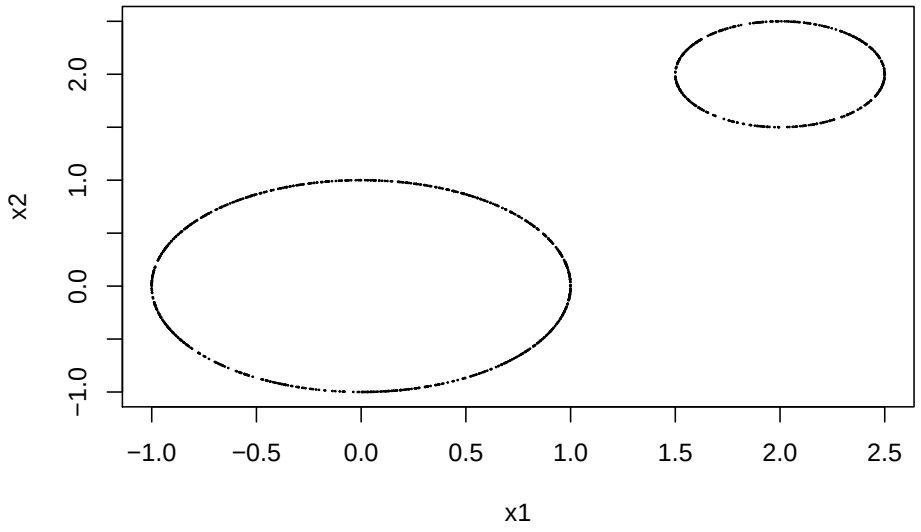
```
x = seq(xlim[1], xlim[2], by = by)
# creating the noisy function values
y = -0.0916 * x^5 + 1.087 * x^4 - 4.53 * x^3 + 7.712 * x^2 - 4.27 * x + 1 +
    rnorm(length(x),mean = 0, sd = 0.04)

# here we calculate the sublevel filtration and the diagram
d = gridDiag(FUNvalues = y, lim = xlim, by = by, maxdimension = 0,
             sublevel = TRUE)
par(mfrow = c(1,2))
plot(x,y, type = 'l')
plot.diagram(d$diagram)
```



2.3.4 R Example: Density Estimator

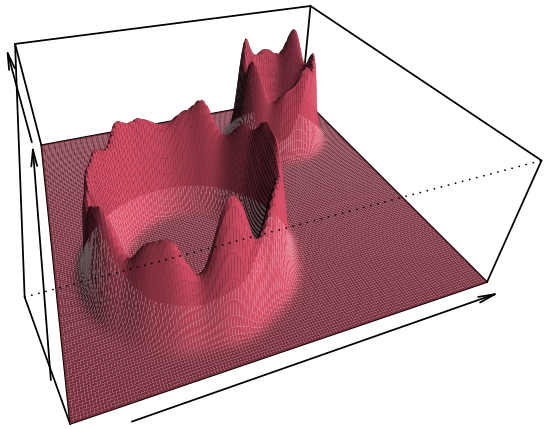
We already saw in the double torus example, that the persistence of a hole depends on its size, when we use a Vietoris-Rips filtration. A different approach of characterizing the topology is by using density estimators and then using superlevel sets in order to analyze the topology. In the example down below we sample two point clouds from two disjoint circles. Afterwards we use a Gaussian kernel density estimator, which is plotted using the `persp()` command and create a persistence diagram of superlevel sets of the estimated density. The diagram shows two red triangles corresponding to the two holes in our data. Allthough the radii differ both holes have nearly the same persistence, since the lifespan now depends on the estimated density. Additionally we plotted a confidence band around the diagonal for 0-dimensional features, i. e. components, using the `bootstrapDiagram()` command which uses the bottleneck distance (see subsection 2.5.1) in order to calculate the confidence interval.

```
# First we sample 800 and 400 points from two circle with radii 1
# and 1/2 respectively
n = 800
c1 = circleUnif(n, r = 1)
c2 = circleUnif(n/2, r = 1/2) + c(2,2)

X = rbind(c1,c2)
plot(X, cex = .1)
```

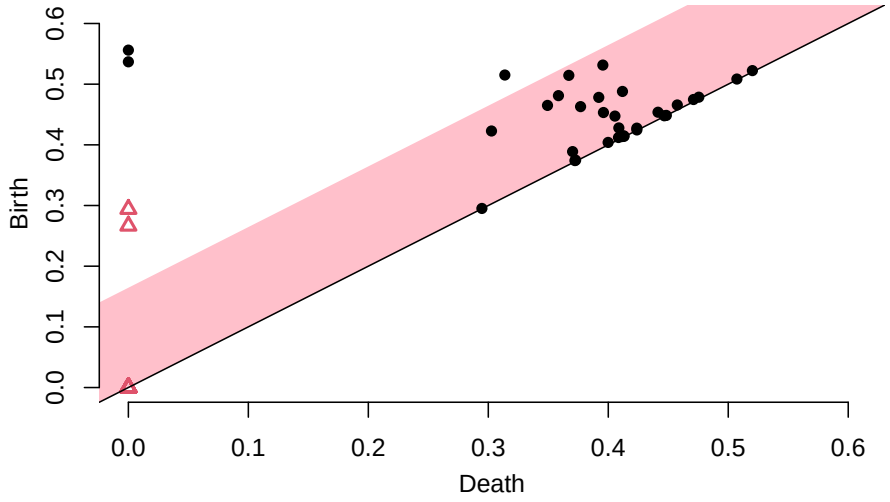



```
# Next we evaluate a kernel density estimator on a grid
Xlim = c(-1.5, 3);
Ylim = c(-1.5, 3);
by = .03

Xseq = seq(Xlim[1],Xlim[2], by = by)
Yseq = seq(Ylim[1],Ylim[2], by = by)
Grid = expand.grid(Xseq,Yseq)
```

```
h = .1
KDE = kde(X = X, Grid = Grid, h = h)

persp(Xseq, Yseq,matrix(KDE, ncol = length(Yseq), nrow = length(Xseq)),
      xlab = "", ylab = "", zlab = "", theta = -20, phi = 35, ltheta = 50,
      col = 2, border = NA, main = "KDE", d = 0.5, scale = FALSE,
      expand = 3, shade = 0.9)
```




```
# Now we calculate the persistence diagram of the superlevel filtration
diag = gridDiag(X = X, FUN = kde, h = h, lim = cbind(Xlim, Ylim), by = by,
                sublevel = FALSE,location = TRUE, printProgress = FALSE)

# And as a last step we calculate a confidence band around the diagonal for
# zero dimensional features, i. e. components, using a bootstrap algorithm

bandComp = bootstrapDiagram(X = X, FUN = kde, h = h, lim = cbind(Xlim, Ylim),
                            by = by, sublevel = FALSE, alpha = 0.05,
                            dimension = 0, distance = "bottleneck", B = 30,
                            printProgress = FALSE)
plot.diagram(diag$diagram, dimension = NULL, band = bandComp,
             diagLim = c(0,max(diag$diagram[,2:3]+.05)))
```



2.4 Analyzing Greyscale Images

As a next application we want to use TDA to analyze the geometry of images. The calculations in this subsection were made with the `C++` software `DIPHA`, see (Reininghaus, Bauer, and Kerber 2014). Given an image D we consider the following function

$$f\colon D \to \{0, \ldots, 255\}$$

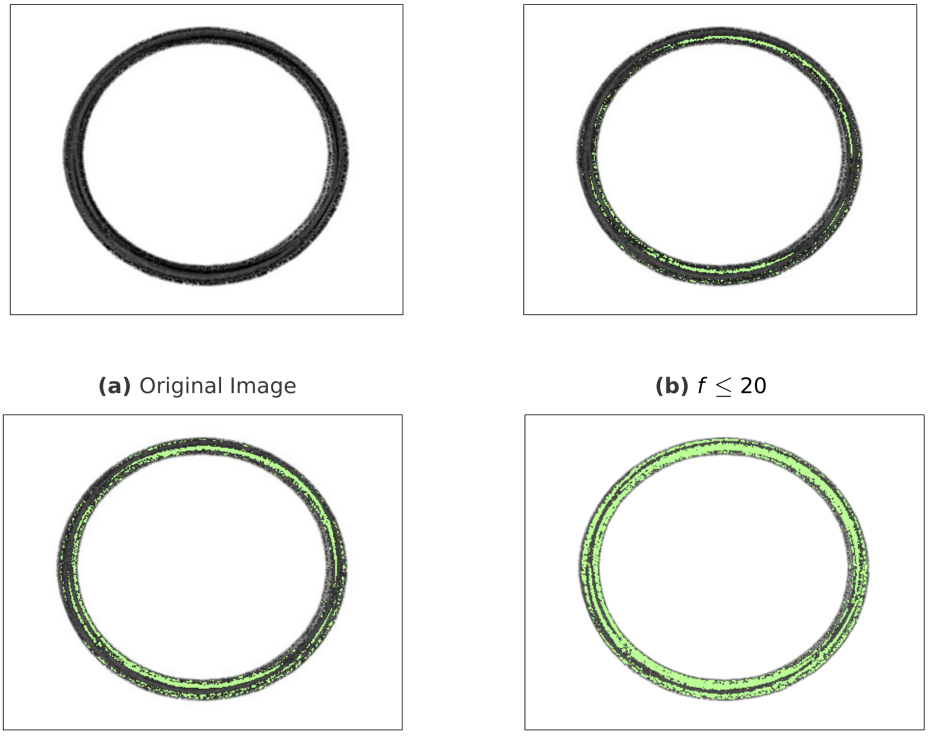
which maps each pixel of the image to its greyscale value and create the sublevel filtration

$$\mathcal{K}_0 \subseteq \ldots \mathcal{K}_{255} = D\,.$$

Since a greyscale value of 0 corresponds to black, $\mathcal{K}_0$ consists of all black pixels and by increasing the grayscale parameter, we subsequently add more and more brighter pixels until we finally add all the white pixels, which results in the complete picture D. This simple filtration still admits the geometric interpretation of counting components, holes, etc.

This is shown in Figure 8. In the upper left hand corner we see the original image of a circle with thick border. Now we generate the filtration and color added pixels in green. In the upper right image, we see the circle where all pixels with a grayscale value smaller or equal than 20 are colored in green. Note how two arches are starting to form. By increasing the value further, more and more pixels are added, until the pixels finally form a circle at a value of 57.

The corresponding persistence diagram of one-dimensional features, i. e. holes, is shown in Figure 9. We see that the actual hole of the annulus is born at 57 and vanishes when we finally add the white pixels at a value of 255. However we see that a lot of noise is generated in this scenario. This is not surprising. Assume for a example a cluster of black pixels, with one slightly brighter pixel in the middle. Obviously this cluster contains a hole in our filtration, however the lifespan of the hole should be very small and thus identified as noise.



(a) Original Image

(b) $f \leq 20$

(c) $f \leq 35$

(d) $f \leq 57$

Figure 8: Visualisation of a part of a sub level filtration of a greyscale image onf an annulus.

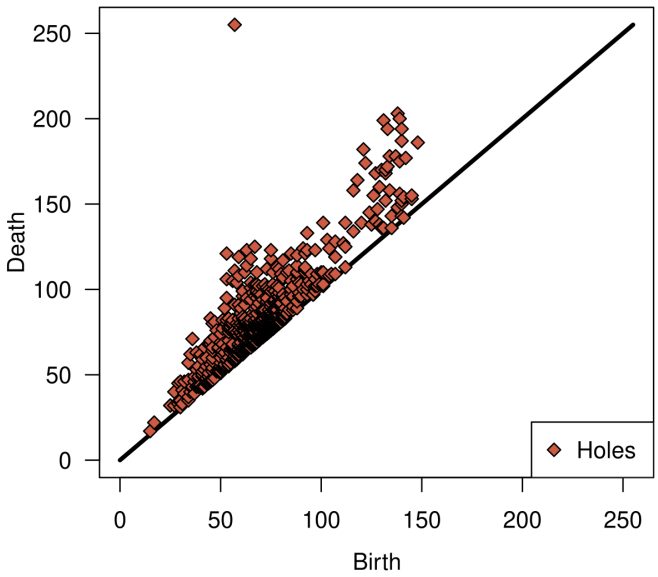


Figure 9: The persistence diagram corresponding to Figure 8.

2.4.1 Real Life Application: Analyzing Fibrin Networks

In (Fenger-Eriksen et al. 2020) we used the above filtration to analyze microscopic images of fibrin nets in order to analyze the effect of different dilutions on patients. Without going into details, we see three images of fibrin networks in Figure 10. The left one is a natural baseline, while the middle and the right ones were spiked with two different doses of some dilution. The corresponding denoised diagrams are shown in Figure 11.

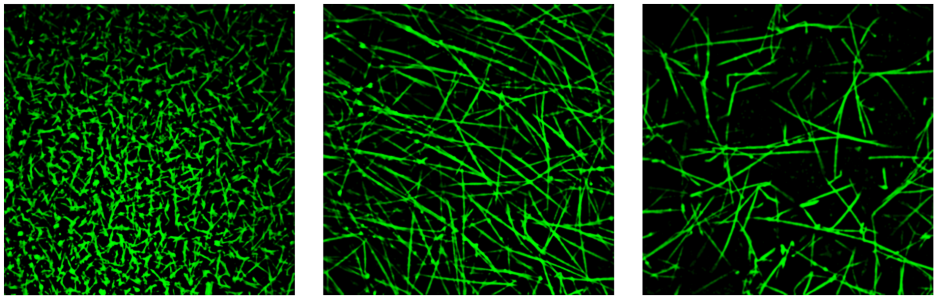


Figure 10: Fibrin nets of a human patient. On the left we see the baseline. The middle and right picture show the net after spiking it with two different doses of a dilution.

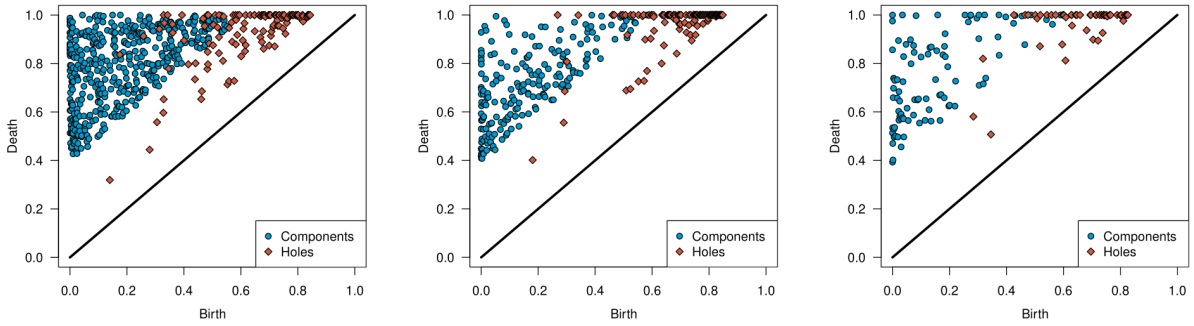


Figure 11: The corresponding denoised persistence diagrams of Figure .

Clearly, we see that the persistence diagrams capture the thinning of the nets.

2.5 Analyzing Persistence Diagrams

We have seen a lot of examples where we can use TDA in order to create persistence diagram, However we still not know how to actually work with them. We now present two (out of many!) methods which are often used to analyze them.

2.5.1 The Metric Space of Persistence Diagrams.

In (Mileyko, Mukherjee, and Harer 2011) it is shown that the space of persistence diagrams forms a metric (more particular polish) space, i. e. we are able to compute the distance between two persistence diagrams. Let $p \geq 1$ then the pth Wasserstein distance of two

persistence diagrams d_1 and d_2 is defined as

$$W_p(d_1, d_2) = \left(\inf_{\sigma} \sum_{x \in d_1} \|x - \sigma(x)\|_\infty^p \right)^{\frac{1}{p}}.$$

where σ ranges over all bijections from d_1 to d_2 and $\|\cdot\|_\infty$ denotes the infinity norm. Considering the case $p = \infty$ we obtain the bottleneck distance

$$d_B(d_1, d_2) = \inf_{\sigma} \max_{x \in d_1} \|x - \sigma(x)\|_\infty .$$

Here we see why we added the diagonal in the definition of a persistence diagram. The Wasserstein distance tries to find the optimal bijection between the two diagrams, which would not be possible if their cardinalities differ. In the `R` package `TDA` we can calculate the distances with the commands `wasserstein()` and `bottleneck()`. Computing these distances is highly non-trivial and can take some time.

2.5.2 Real Life Application: Analyzing Fibrin Networks Pt. 2

In our fibrin nets example, we calculated the distances between the obtained diagrams and the empty diagram for several patients. In Figure 12you see in the first three columns the obtained means and down below the 95% confidence intervall, of distances to the empty diagram of the Baseline, the first spiking step (S1) and the second spiking step (S2) for holes and components. The fourth and fith column show the obtained p values when applying a two-sided t-Test. As you can see, we obtain significant results for holes.

				Baseline - S1	Baseline - S2
Wasserstein distance	**Baseline**	**S1**	**S2**	**p value**	**p value**
Components	144.82 (120.21 to 169.42)	109.21 (87.53 to 130.88)	101.91 (81.61 to 122.22)	$p = 0.012$	$p = 0.016$
Holes	20.81 (16.62 to 25.01)	17.39 (13.82 to 20.95)	17.03 (13.17 to 20.89)	$p = 0.093$	$p = 0.042$

Figure 12: Wasserstein distances from the diagonal; p values.

2.5.3 Power-weighted Silhouettes

Another method to analyze the topology of the data, are power-weighted silhouettes given in (Chazal, Fasy, et al. 2013). Assume a persistence diagram with off diagonal points $(b_j, d_j)_{j=1}^N$ for $N \in \mathbb{N}$. Then we define the hat functions via

$$\Lambda_{p_j}(t) = \begin{cases} t - b_j, & \text{if } t \in \left[b_j, \frac{b_j + d_j}{2}\right] \\ d_j - t, & \text{if } t \in \left(\frac{b_j + d_j}{2}, d_j\right] \\ 0, & \text{else} \end{cases}$$

where $p_j = (b_j, d_j)$ for $j = 1, \ldots, N$. Using the above functions we can define for $0 < p < \infty$ the **power-weighted silhouette** as

$$\phi^{(p)}(t) = \frac{\sum_{j=1}^N |d_j - b_j|^p \Lambda_{p_j}(t)}{\sum_{j=1}^N |d_j - b_j|^p}$$

for $t \in \mathbb{R}$. So a silhouette is just a weighted sum of the above hat functions. The parameter p weighs the persistence of each point. When p is small the silhouette puts more emphasis on points with lower lifespan while a larger p gives more attention to points with higher persistence. In Figure 13 we see an example of a silhouette corresponding to a diagram in Figure 11. The command in the `TDA` package for creating such silhouettes is `silhouette()`.

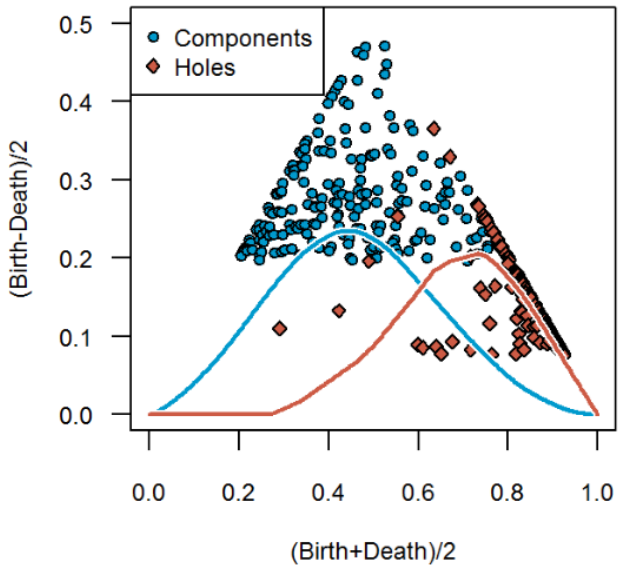


Figure 13: Weighted silhouette of a persistence diagram in Figure 11 for components as well as holes.

2.6 Clustering

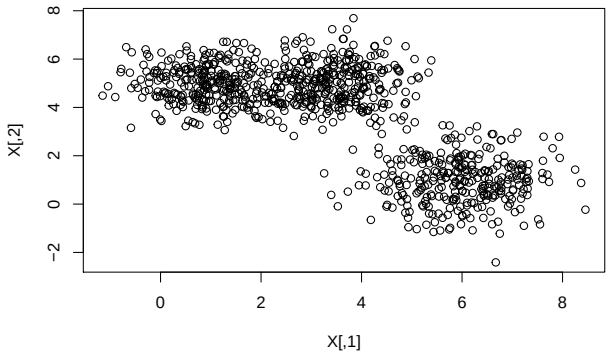
As final topic we want to briefly discuss clustering with topological data analysis. A nice cluster algorithm, called *Tomato Clustering* can be found in (Chazal, Guibas, et al. 2013). However, we will only briefly cover the framework implemented in the `R` package `TDA`. The package provides an implementation of Density-Based Clustering (DeBaCl) given in (Kent, Rinaldo, and Verstynen 2013). Assume the following scenario. We are given a point clound $X \subset \mathbb{R}^d$ which we assume to be sampled i. i. d. from an unknown probability distribution with density function f. Then DeBaCl provides a procedure to find clusters in our point cloud without any a priori knowledge of f and the number of clusters by using superlevel sets. Using the notation of the `TDA` package documentation and [kent2013debacl] we define for $\lambda \geq 0$

$$L_\lambda(f) = \{x \in \mathbb{R}^d \,|\, f(x) \geq \lambda\}\,.$$

The connected components of $L_\lambda(f)$ are called λ-clusters of f. Again, the idea is now to increase the value of λ and keep track of the occuring connected components. Since λ is getting bigger, obviously points can fall out of the the superlevel set. Assume a point x drops out of the set. This yields three situations: In the first case, basically nothing changes, i. e. the clusters remain the same. In the second scenario, x was the connecting link between two clusters, thus a cluster will split into two different connected components. In the third

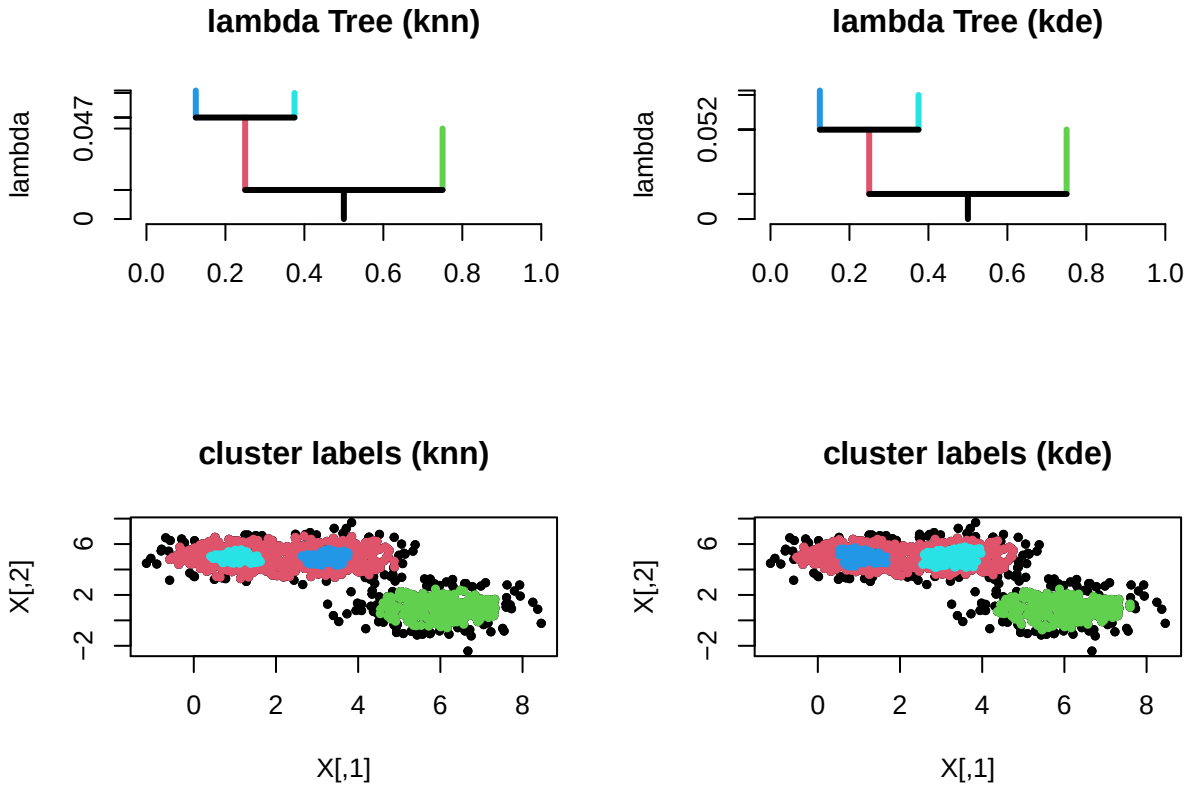
case the point x might have been the last point of a connected component, thus by kicking x out of the superlevel set, a cluster vanishes. By taking track of all this scenarios, we should obtain a good idea of the connected components, allowing us to cluster the dataset. However, we actually need f to compute the superlevel set. Since f is not known, we need to estimate it. We don't discuss the procedure of approximating f here, however the `TDA` package provides the option to specify some density estimaters like gaussian kernel functions or nearest neighbours estimators for estimating f. Having estimated f to obtain $L_\lambda(f)$ the package then uses a nearest neighbours approach to estimate the clusters. Let's have a look at the example given in the documentation of the `TDA` package.

```
# First we generate three not well separated clusters
X1 = cbind(rnorm(300, 1, .8), rnorm(300, 5, 0.8))
X2 = cbind(rnorm(300, 3.5, .8), rnorm(300, 5, 0.8))
X3 = cbind(rnorm(300, 6, 1), rnorm(300, 1, 1))

X = rbind(X1, X2, X3)
plot(X)
```




```
# Here we calculate the appearing clusters using a nearest neighbour and
# gaussian kernel approach. Note that even in the Gaussian kernel setting
# we need to specify k = 100 for the nearest neighbour approach used to
# calculate the clusters in the superlevel sets
TreeKNN = clusterTree(X, k = 100, density = "knn", printProgress = FALSE)
TreeKDE = clusterTree(X, k = 100, h = 0.3, density = "kde",
                      printProgress = FALSE)

# Here we plot the computed cluster trees (see below)
par(mfrow = c(2,2))
plot(TreeKNN, type = "lambda", main = "lambda Tree (knn)")
plot(TreeKDE, type = "lambda", main = "lambda Tree (kde)")

# Here we plot the clusters
# plot clusters
plot(X, pch = 19, cex = 0.6, main = "cluster labels (knn)")
for (i in TreeKNN[["id"]]){
  points(matrix(X[TreeKNN[["DataPoints"]][[i]],],ncol = 2), col = i,
```

```
          pch = 19, cex = 0.6)
}

plot(X, pch = 19, cex = 0.6, main = "cluster labels (kde)")
for (i in TreeKDE[["id"]]){
  points(matrix(X[TreeKDE[["DataPoints"]][[i]],],ncol = 2), col = i,
          pch = 19,cex = 0.6)
}
```



In the above example we generated three clusters and applied the above procedure using two different density methods. Looking at the plots of the clusters, both obtain quite similar results. Additionally we also plot the cluster trees. Cluster trees provide a nice illustration of the calculations of the above algorithm and also allow us to somehow get a feeling of the geometry of higher dimensional datasets. The interpretation is as follows. The y-axis corresponds to the value of λ and each verticle line represents a cluster. If at certain λ a cluster splits into new connected components, the vertical line forks into new vertical lines corresponding to the new clusters. If a cluster completely vanishes at a certain level, the vertical line stops at this value. Looking at the obtained cluster trees in the above example we see the following. At $\lambda = 0$ all points lie in the superlevel set, and apparently when we use a knn approach for clustering, we obtain one cluster which corresponds to the first vertical line, i.e. the vertical line starting with y coordinate zero. Then this connected component splits into two new clusters (red and green). One of these clusters vanishes at a certain density level (green) while the other splits again into two new clusters (blue and light blue), which do not split further and vanish at a high enough density level, resulting into three high

density clusters.

3 Conclusion

The goal of the above notes was to give an introduction to topological data analysis. We discussed the geometric ideas of topology and afterwards introduced the fundamental notion of filtrations and discussed a way of visualizing a filtration by using persistence diagrams. We also observed two popular methods to furhter statistically analyze these diagrams, namely the wasserstein/bottleneck distances and weighted silhouettes. Furthermore we discussed how to apply these methods to grayscale images and then used filtrations to cluster datasets. While these toy examples might look like very specific problems, keep in mind that the ideas are very general. As soon as you have a distance function, you can apply TDA with the geometric interpretation we discussed at the beginning. But a metric is not mandatory. The framework of sub and superlevel sets allows for basically arbitrary filtrations. However TDA alone will often not be enough to completely solve a statistical problem. Rather it should be considered as an additional feature, which might help in understanding the data.

4 Exercises

Exercise 1

Sample a point cloud from two different disjoint circles and plot a persistence diagram using a superlevel filtration of a density estimator of your choice and also plot the density estimator. Now add noise all over the plane (see Figure 14) and repeat the above. Calculate the wasserstein distance for $p = 1$ between the two diagrams and plot a silhouette for each diagram, for components as well as holes.

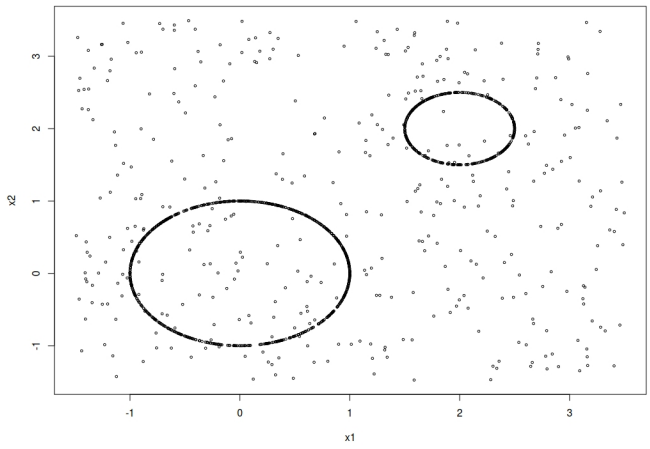


Figure 14: Point cloud sampled from two circles with noise.

Exercise 2

Sample a point cloud similar to the one shown in Figure 15 and try to cluster it. (Don't expect perfect results, it's about playing around with the package.) *Hint:* Every point on the unit circle around the origin has the form $(\cos(\varphi), \sin(\varphi))$ for $\varphi \in [0, 2\pi)$. Use this parametrization and restrict the angle to obtain the desired point cloud.

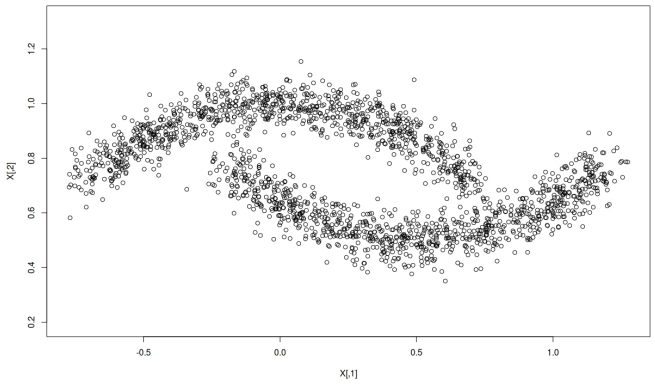


Figure 15: A fancy point cloud.

Bibliography

Chazal, Frédéric, Brittany Terese Fasy, Fabrizio Lecci, Alessandro Rinaldo, and Larry Wasserman. 2013. "Stochastic Convergence of Persistence Landscapes and Silhouettes." http://arxiv.org/abs/1312.0308.

Chazal, Frédéric, Leonidas J. Guibas, Steve Y. Oudot, and Primoz Skraba. 2013. "Persistence-Based Clustering in Riemannian Manifolds." *J. ACM* 60 (6). https://doi.org/10.1145/2535927.

Fasy, Brittany Terese, Jisu Kim, Fabrizio Lecci, and Clément Maria. 2014. "Introduction to the R Package TDA." *CoRR* abs/1411.1830. http://arxiv.org/abs/1411.1830.

Fenger-Eriksen, C., AD Lindholm, L. Krogh, T. Hell, M. Berger, M. Hermann, D. Fries, N. Juul, M. Rasmussen, and AM Hvas. 2020. "Effect of Tranexamic Acid on Coagulation and Fibrin Clot Properties in Children Undergoing Craniofacial Surgery." *Thrombosis and Haemostasis*.

Gholizadeh, Shafie, and Wlodek Zadrozny. 2018. "A Short Survey of Topological Data Analysis in Time Series and Systems Analysis." *CoRR* abs/1809.10745. http://arxiv.org/abs/1809.10745.

Kent, Brian P., Alessandro Rinaldo, and Timothy Verstynen. 2013. "DeBaCl: A Python Package for Interactive Density-Based Clustering." http://arxiv.org/abs/1307.8136.

Li, L., W. Y. Cheng, B. S. Glicksberg, O. Gottesman, R. Tamler, R. Chen, E. P. Bottinger, and J. T. Dudley. 2015. "Identification of type 2 diabetes subgroups through topological

analysis of patient similarity." *Sci Transl Med* 7 (311): 311ra174.

Mileyko, Yuriy, Sayan Mukherjee, and John Harer. 2011. "Probability Measures on the Space of Persistence Diagrams." *Inverse Problems* 27 (12): 124007. https://doi.org/10.1088/0266-5611/27/12/124007.

Nicolau, M., A. J. Levine, and G. Carlsson. 2011. "Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival." *Proc. Natl. Acad. Sci. U.S.A.* 108 (17): 7265–70.

Ravishanker, Nalini, and Renjie Chen. 2019. "Topological Data Analysis (Tda) for Time Series." http://arxiv.org/abs/1909.10604.

Reininghaus, J., U. Bauer, and M. Kerber. 2014. "DIPHA, a High Performance C++ Software Package for Distributed Topological Data Analysis Using Persistent Homology." https://github.com/DIPHA/dipha.