Chapter 7

Reeb Graphs and Mapper

In this chapter we look at another tool in topological data analysis, called *Mapper*. The underlying idea of Mapper has its roots in Morse theory, where Georges Reeb defined a graph to summarize a Morse function on a manifold. We first discuss these graphs, called *Reeb graphs*, and then how to mimic the ideas for the case where instead of a manifold we have point cloud data.

Before we dive into the mathematical details, a short remark about the pronunciation of the word "Reeb graph": Georges Reeb, after whom these graphs are named, was a French mathematician born in the German speaking region Alsace. Thus, he likely pronounced his name the German way, that is, with the "ee" spoken similar to the "ea" in "bear" (as opposed to "beer").

7.1 Reeb Graphs

The idea of Reeb graphs is that given some topological space X, and some function $f : X \to \mathbb{R}$, we consider the preimage of f for some fixed value $a \in \mathbb{R}$. We place one point per connected path-component of the preimage. We do this for all values in $\mathbb{R}$, and connect the points corresponding to neighboring connected components in adjacent preimages. More formally,

Definition 7.1. *Let X be some topological space, and f a function $f : X \to \mathbb{R}$. Two points x, y are called equivalent ($x \sim y$), iff $f(x) = f(y) = \alpha$ and x and y are in the same path-connected component of $f^{-1}(\alpha)$. The* Reeb graph R_f *is the quotient space $X/\sim$.*

We assume all of our functions to be *levelset tame* for the space X:

Definition 7.2. *A function $f : X \to \mathbb{R}$ is* levelset tame *if*

- *each levelset $f^{-1}(\alpha)$ has finitely many connected components, all of which are path-connected, and*

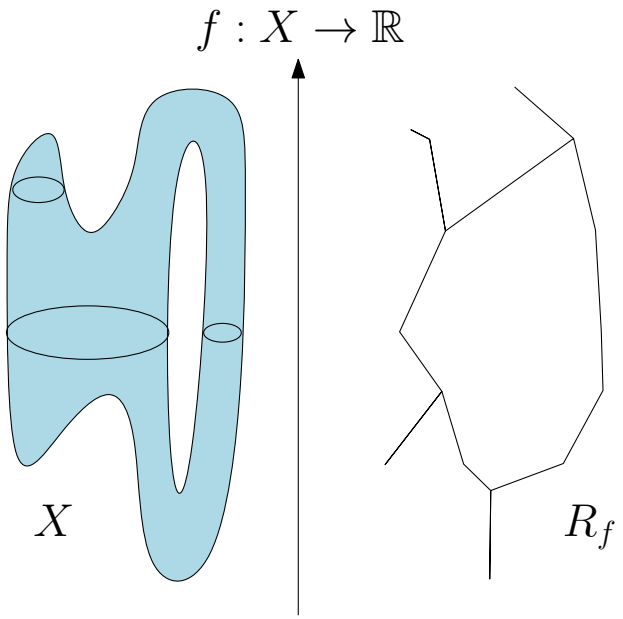

Figure 7.1: *An example of a Reeb graph*

- *the homology groups of the levelsets only change at finitely many critical values.*

As we have defined it, the Reeb graph is just a (continuous) topological space. We call it a graph since it is 1-dimensional, but to arrive at a graph as we know it in combinatorics, we will need to discretize it. For this we need to define vertices and edges. There are many different possibilities of defining vertices and edges to discretize the Reeb graph, but we want to define some type of minimal one.

Let us look at the neighborhood of some point p in the Reeb graph (as a topological space). We look at how many ways there exist to go from p towards the direction of higher f-value (we call this number the up-degree u), and how many ways to go towards the direction of lower f-value (we call this the down-degree l). Depending on u and l, we classify p as in Table 7.1.

Table 7.1: *Classifications of points in the Reeb graph.*

u	l	Classification
1	1	regular
0	> 0	maximum
> 0	0	minimum
$\geqslant 2$	l	up-fork
u	$\geqslant 2$	down-fork

Note that a point can fall into multiple of these classes, for example it can be a maximum and a down-fork simultaneously, or an up-fork and a down-fork simultaneously. We

call the minima, maxima, up-forks, and down-forks *critical points*. Our discretization places vertices at the critical points, and all the connections between critical points (that are made up of only regular points) become the edges of our graph. Note that the graph we get through this process is not necessarily simple, we may have multi-edges.

Exercise 7.3. *Consider a double torus embedded in* $\mathbb{R}^3$*. You can imagine it as the result of taking the figure depicted in Figure 7.2 embedded in the plane* $x_3 = 0$*, replacing every point by a 3-dimensional ball with radius* $r < \min\{d/2, R/2\}$*, and taking the boundary of the union of these balls.*

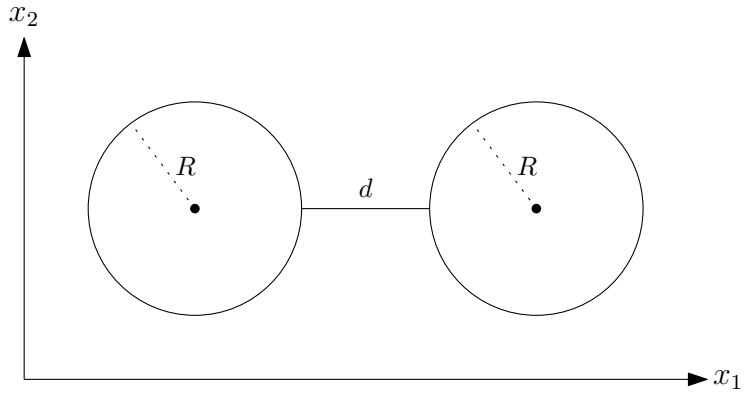

Figure 7.2: *The space blown up to a double torus in Exercise 7.3.*

Draw the Reeb graphs for the functions $f_1(x) = x_1$*,* $f_2(x) = x_2$*, and* $f_3(x) = x_3$*.*

We next consider merge trees and split trees, which are variants of the Reeb graph, where instead of levelsets, we look at sub-level sets or super-level sets.

Definition 7.4. *Let* X *be some topological space, and* f *a function* $f : X \to \mathbb{R}$*. We have* $x \sim_M y$ *for two points* x, y*, if and only if* $f(x) = f(y) = \alpha$ *and* x *and* y *are in the same connected component of* $f^{-1}((-\infty, \alpha])$*. The* merge tree T_M *is the quotient space* $X/\sim_M$*.*

Note that in the merge tree, since we only increase the space under consideration, we never have a connected component that splits. We can only have new connected components appearing, and connected components merging. This also tells us that the Merge tree (or its discretization) is always a tree.

Definition 7.5. *Let* X *be some topological space, and* f *a function* $f : X \to \mathbb{R}$*. We have* $x \sim_S y$ *for two points* x, y*, if and only if* $f(x) = f(y) = \alpha$ *and* x *and* y *are in the same connected component of* $f^{-1}([\alpha, \infty))$*. The* split tree T_S *is the quotient space* $X/\sim_S$*.*

To be able to compute Reeb graphs in actual TDA applications, we now look at Reeb graphs in the context of simplicial complexes. We consider a simplicial complex K and a function $f : |K| \to \mathbb{R}$, which is piece-wise linear (linear on each simplex). We observe that

the Reeb graph then only depends on the 2-skeleton of K. This is the case since looking at a levelset is the same as cutting through the simplicial complex. When we cut through a simplex, we generally get a simplex of one dimension lower. In a simplicial complex, connectivity is completely determined by the 1-skeleton. Thus, before cutting, the 2-skeleton suffices. We can also see that the critical points are images of the vertices of K. This happens since a connected component can only appear, disappear, split, or merge at some local maximum or minimum of the connected component. Since the function is linear, the maximum or minimum of every simplex is always attained at some vertex. We define the *augmented Reeb graph* of a simplicial complex with a PL-function, as the discretization of the Reeb graph by placing a vertex for every connected component in all the levelsets for the values a for which $f(v) = a$ for some vertex of the simplicial complex.

Theorem 7.6. *Given a simplicial complex* K *with* m *faces and a piece-wise linear function* $f : |K| \to \mathbb{R}$ *on it, we can compute the augmented Reeb graph* R_f *of* K *with respect to* f *in time* $O(m \log m)$.

Proof. We only have to consider the 2-skeleton of K. To compute the augmented Reeb graph, we perform a discrete sweep (or scan) through K in the order given by f, only stopping at values a such that $f(v) = a$ for some vertex v. In this sweep, we want to keep track of the connected components.

For any $\alpha \in \mathbb{R}$, the levelset $f^{-1}(\alpha)$ of the 2-skeleton of K is just a graph G_α: vertices and edges of K induce vertices of G_α, triangles induce edges. In the open interval between any two values a with $f(v) = a$, G_α stays the same. In our sweep, we track the graph G_α. Whenever we enter a vertex v, faces may end: The vertices in G_α that correspond to edges vw in K with $f(v) > f(w)$ all get merged into a single vertex. Whenever we then exit v, some faces may start. The vertex corresponding to v is split into one vertex per edge vw in K such that $f(v) < f(w)$. Some of these vertices are connected: the vertices corresponding to the edges vw and vx are connected by an edge if vwx is a triangle of K.

To build the augmented Reeb graph out of these G_α graphs, we use a dynamic spanning forest data structure. Such a data structure can update connected components under both insertion and deletions of edges (vertices we only delete when also deleting all their incident edges). There exist such data structures that can do each update in amortized time $O(\log m)$, where m is the size of the graph. The size of the graph is bounded by the sum m of vertices, edges, and triangles in K. Each such feature appears at one point, and disappears at one point, and we thus have at most $2m$ insertions and deletions in total, giving an $O(m \log m)$ algorithm. □

Exercise 7.7. *Consider a simplicial complex* K *and a PL (piece-wise linear) function* $f : |K| \to \mathbb{R}$. *What happens to the Reeb graph when you add one additional face to* K *and extend* f *accordingly?*

7.1.1 Homology of Reeb Graphs

The Reeb graph of a topological space X with respect to a function f can be viewed as a summary of X through the lens of f. The natural question is: how good of a summary is it? It is clear that in general we lose information, for example on the dimension of X, but we can still hope that some topological information is retained. In this section, we thus compare the homology of the Reeb graph to the homology of X. Since the Reeb graph R_f is a graph (a 1-dimensional object), we have $H_p(R_f) = 0$ for $p \geqslant 2$, so any higher-dimensional homology gets lost. However, a graph can still have non-trivial homology in dimensions 0 and 1.

Observation 7.8. *For a levelset tame* $f : X \to \mathbb{R}$*, we have* $\beta_0(X) = \beta_0(R_f)$.

In other words, the Reeb graph captures the 0-homology of the input space X perfectly, no matter which levelset tame function f we use.

Sadly, the same does not hold for the 1-homology. Let us consider a torus, as in Figure 7.3. In general, it can be that the choice of function f determines whether we capture a hole or not, consider for example a cylinder. However note that for the torus, it is actually the case that no matter which function f we choose, we cannot capture its 1-homology (this is non-trivial to show).

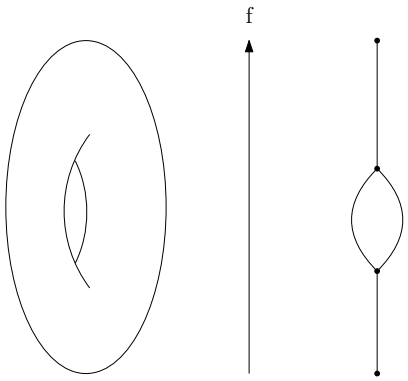

Figure 7.3: *The torus and its Reeb graph.*

On the other hand, we can see that every cycle in the Reeb graph is indeed also a cycle in the topological space X, and it cannot be filled in, so it is indeed a hole. Thus we also get the following observation:

Observation 7.9. *For a levelset tame* $f : X \to \mathbb{R}$*, we have* $\beta_1(X) \geqslant \beta_1(R_f)$.

Can we somehow formalize which holes we lose? To do this, we split up homology into "horizontal" and a "vertical" parts. The intuition behind these terms is that we consider f to be a function pointing upwards, and thus roughly, horizontal means "perpendicular to f" and vertical means "along f".

Definition 7.10. *A p-th homology class $h \in H_p(X)$ is called* horizontal *if there is a finite set of values $A = \{a_1, \ldots, a_k\}$ such that $h \in \mathrm{im}(\iota_*)$ where ι_* is the map $H_p(\bigcup_{a \in A} X_a) \to H_p(X)$ induced by inclusion, where $X_a = f^{-1}(a)$.*

This definition means that we need to be able to find a finite set of levelsets, such that we can find cycles contained in these levelsets, which are in the homology class h in $H_p(X)$.

One now wonders whether the set of horizontal homology classes forms a group. Let this set be $\overline{H_p}(X)$. It turns out that it is indeed a group.

Lemma 7.11. *$\overline{H_p}(X)$ is a subgroup of $H_p(X)$.*

Proof. First, we see that the identity element 0 is in $\overline{H_p}(X)$. We can take an arbitrary set A, and we can always map the 0 element of $H_p(\bigcup_{a \in A} X_a)$ to 0.

Next, we show that the set is closed under addition. Let $p, q \in \overline{H_p}(X)$, and we show that $p + q \in \overline{H_p}(X)$. p has a pre-image in some levelset A_p, and q has a pre-image in some levelset A_q. $p + q$ must have a pre-image in $A_p \cup A_q$.

Finally, we show that the inverse of every element is contained in the group, but since every element is self-inverse in $\mathbb{Z}_2$-homology, we get that for every element its inverse is also contained in $\overline{H_p}(X)$. □

Since the horizontal homology is a sub-group, we can now easily define *vertical homology* by taking quotient groups.

Definition 7.12. *The* vertical homology group *of X with respect to f is the group $\overset{\vee}{H}_p(X) := H_p(X)/\overline{H_p}(X)$.*

Observation 7.13. $\dim(H_p(X)) = \dim(\overline{H_p}(X)) + \dim(\overset{\vee}{H}_p(X))$.

The following fact that we do not prove here shows that when we go from a space X to its Reeb graph, we keep the vertical homology classes, and lose the horizontal ones.

Fact 7.14. *The surjection $\phi : X \to R_f$ induces an isomorphism $\overset{\vee}{\Phi} : \overset{\vee}{H}_1(X) \to H_1(R_f)$.*

Corollary 7.15. *Given X an orientable connected compact 2-manifold, and a Morse function $f : X \to \mathbb{R}$, then $\beta_1(R_f) = \beta_1(X)/2$.*

Here, a 2-manifold is a space that locally at every point looks like $\mathbb{R}^2$. Orientable means that there is an inside and an outside side. A Morse function is a "nice enough" function defined in terms of some derivatives, which we do not need to specify here.

Exercise 7.16. *(a) Consider a 2-dimensional geometric simplicial complex K embedded in $\mathbb{R}^2$. Consider the function $f(x) = x_1$. Show that $\beta_1(K) = \beta_1(R_f)$.*

(b) Find a geometric simplicial complex K embedded in $\mathbb{R}^2$ and a map $f : K \to \mathbb{R}$ such that $\beta_1(K) > \beta_1(R_f)$.

7.2 Distances for Reeb Graphs

In order to compare Reeb graphs to each other, we again want to define distance measures between them. We discuss two such measures here. The first one, called *interleaving distance*, is, not surprisingly, similar to the interleaving distance of persistence modules. The second one, called *functional distortion distance* is similar to the Gromov-Hausdorff distance for metric spaces.

7.2.1 Interleaving Distance

When do we want two Reeb graphs to be considered the same, and thus have distance 0? We definitely need that the graphs are isomorphic in the sense of graph isomorphism. But further than that, we also want that this graph isomorphism is "function preserving". In other words, the critical points should lie on the same function levels. The idea of the interleaving distance is to measure how far away from this we are. Thus, given two Reeb graphs R_f, R_g, "how much" is missing towards a "function preserving isomorphism"? Towards formalizing this idea, we need a few definitions.

Note that when we compare two Reeb graphs R_f, R_g, those can be Reeb graphs of *different spaces* with regards to *different functions*.

Definition 7.17. *An* ϵ-thickening X_ϵ *of some topological space* X *is given by* $X_\epsilon := X \times [-\epsilon, +\epsilon]$.

Definition 7.18. *For a Reeb graph* R_f *consider a function* $f_\epsilon : (R_f)_\epsilon \to \mathbb{R}$ *such that*

$$(x, t) \mapsto f(x) + t.$$

The ϵ-smoothing *of* R_f*, denoted by* $S_\epsilon(R_f)$ *is the Reeb graph of* $(R_f)_\epsilon$ *with regards to* f_ϵ.

An example of these definitions can be seen in Figure 7.4. Note that when we say $(R_f)_\epsilon$, we mean an ϵ-thickening of R_f, *not* a Reeb graph with regards to some function f_ϵ. The ϵ-smoothing $S_\epsilon(R_f)$ is then a Reeb graph with regards to the function f_ϵ, but *of* $(R_f)_\epsilon$, and not of the original space that R_f is the Reeb graph of. Furthermore, when we write $f(x)$ for some $x \in R_f$, we mean that we extend f to some function $f^* : R_f \to \mathbb{R}$, simply by mapping x (an element of $X/\sim$, i.e., an equivalence class of $\sim$) to $f(p)$ for some point $p \in X$ in the equivalence class x. We will just call this function f as well for simplicity.

Definition 7.19. *The function* $\iota : R_f \to S_\epsilon(R_f)$ *with* $x \mapsto [(x, 0)]$ *is the* quotiented inclusion map*. Here,* $[(x, 0)]$ *denotes the equivalence class, or the connected component that contains* $(x, 0)$ *in* $f_\epsilon^{-1}(f_\epsilon(x, 0))$.

Consider some function $\mu : R_f \to R_g$ which is function preserving, i.e., $f(x) = g(\mu(x))$ for all $x \in R_f$. A function-preserving map $\mu : R_f \to S_\epsilon(R_g)$ induces a function preserving map $\mu_\epsilon : S_\epsilon(R_f) \to S_{2\epsilon}(R_g)$ with $[(x, t)] \mapsto [(\mu(x), t)]$.

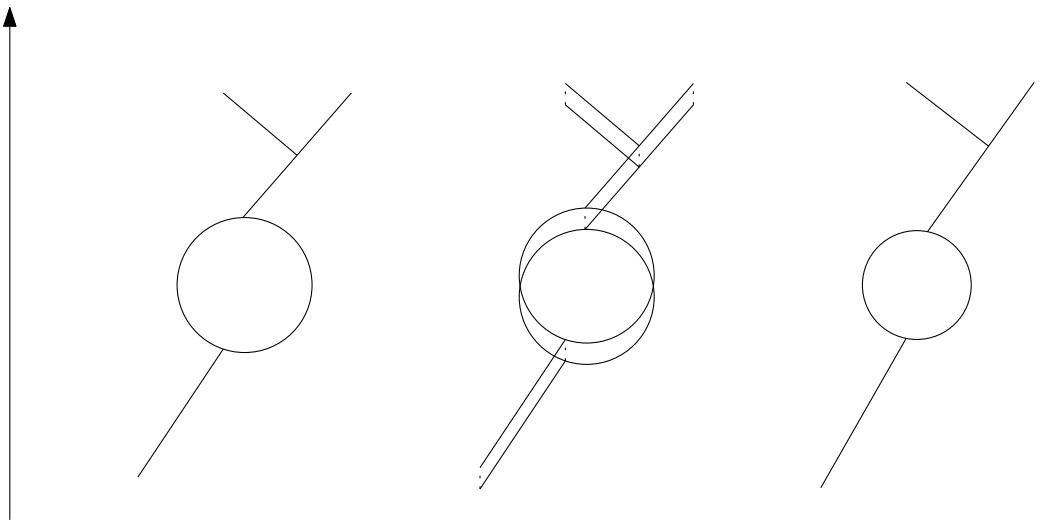

Figure 7.4: *A Reeb graph, its ϵ-thickening, and its ϵ-smoothing.*

Definition 7.20 (Reeb graph interleaving). *Two Reeb graphs R_f, R_g are* ϵ-interleaved*, if there exists a pair of function preserving maps $\varphi : R_f \to S_\epsilon(R_g)$, $\psi : R_g \to S_\epsilon(R_f)$ such that the following diagram commutes:*

$$\begin{array}{ccccc} R_f & \xrightarrow{\iota} & S_\epsilon(R_f) & \xrightarrow{\iota_\epsilon} & S_{2\epsilon}(R_f) \\ & \searrow^{\psi} \quad \nearrow_{\varphi} & & \searrow^{\psi_\epsilon} \quad \nearrow_{\varphi_\epsilon} & \\ R_g & \xrightarrow{\iota} & S_\epsilon(R_g) & \xrightarrow{\iota_\epsilon} & S_{2\epsilon}(R_g) \end{array}$$

Here, to understand why ι_ϵ makes sense, we need the following fact, the proof of which is left as an exercise.

Observation 7.21. $S_\delta(S_\epsilon(R_f)) \cong S_{\delta+\epsilon}(R_f)$.

Note that by construction of ι, ι_ϵ and φ_ϵ (or ψ_ϵ, respectively), the trapezoidal parts of this diagram commute trivially: $\varphi_\epsilon \circ \iota(x) = \varphi_\epsilon([(x,0)]) = [(\varphi(x),0)] = \iota_\epsilon \circ \varphi(x)$. Furthermore, any two compact and connected Reeb graphs are ϵ-interleaved for some ϵ. Lastly, if R_f and R_g are ϵ-interleaved, then they are also δ-interleaved for all $\delta \geqslant \epsilon$.

Definition 7.22. $d_I(R_f, R_g) = \inf\{\epsilon \mid R_f, R_g \text{ are } \epsilon\text{-interleaved}\}$.

We once again have a stability theorem, which we will not prove here.

Theorem 7.23. *For tame functions $f, g : X \to \mathbb{R}$ we have $d_I(R_f, R_g) \leqslant \|f - g\|_\infty$.*

7.2.2 Functional Distortion Distance

As mentioned above, the functional distortion distance is motivated by the Gromov-Hausdorff distance for metric spaces. Thus, the first step is to define a metric on a single Reeb graph.

Definition 7.24. *Let R_f be a Reeb graph of a space X, and $u, v \in R_f$ (in the same connected component of R_f), and let π be a path from u to v. We define the* height *of π as $\text{height}(\pi) = \max_{x\in\pi} f(x) - \min_{x\in\pi} f(x)$. To turn this into a distance metric, we consider $\Pi(u, v)$, the set of all paths between u and v. Then, the* function induced metric *on R_f is defined as*

$$d_f(u, v) = \min_{\pi\in\Pi(u,v)} \text{height}(\pi).$$

In a sense, $d_f(u, v)$ is the "thickness" of the thinnest "slice" of the space X in which u and v are connected.

Definition 7.25 (Functional distortion distance). *Let R_f and R_g be two Reeb graphs. Let $\Phi : R_f \to R_g$, $\Psi : R_g \to R_f$ be continuous functions, but not necessarily function-preserving. Then, we define correspondence and distortion:*

$$C(\Phi, \Psi) = \{(x, y) \in R_f \times R_g \mid \Phi(x) = y \text{ or } x = \Psi(y)\}$$

$$D(\Phi, \Psi) = \sup_{(x,y),(x',y')\in C(\Phi,\Psi)} \frac{1}{2}|d_f(x, x') - d_g(y, y')|.$$

And finally, we define the functional distortion distance,

$$d_{FD}(R_f, R_g) = \inf_{\Phi,\Psi} \max\{D(\Phi, \Psi), \|f - (g \circ \Phi)\|_\infty, \|g - (f \circ \Psi)\|_\infty\}.$$

Also for this distance measure there is a stability theorem.

Theorem 7.26. *Let $f, g : X \to \mathbb{R}$ be tame functions. Then, $d_{FD}(R_f, R_g) \leqslant \|f - g\|_\infty$.*

We can also quantify the relation between the two discussed distances.

Theorem 7.27. $d_I(R_f, R_g) \leqslant d_{FD}(R_f, R_g) \leqslant 3d_I(R_f, R_g)$.

Exercise 7.28. *Consider a merge tree T with regards to a function f. We define the a-shift x^a for any $x \in T$ to be the unique "ancestor" of x with function value $f(x^a) = f(x) + a$.*

We now consider two merge trees; T_1 with regards to f, and T_2 with regards to g. We call T_1 and T_2 ϵ-compatible *if there exist maps $\alpha : T_1 \to T_2$ and $\beta : T_2 \to T_1$ such that we get the following commutativities:*

- $g(\alpha(x)) = f(x) + \epsilon$ *for all* $x \in T_1$

- $f(\beta(y)) = g(y) + \epsilon$ *for all* $y \in T_2$
- $\beta \circ \alpha(x) = x^{2\epsilon}$ *for all* $x \in T_1$
- $\alpha \circ \beta(y) = y^{2\epsilon}$ *for all* $y \in T_2$.

The interleaving distance between merge trees $d_I(T_1, T_2)$ *can now be defined as the infimum of all* ϵ *such that* T_1 *and* T_2 *are* ϵ*-compatible. Show that* $d_I(T_1, T_2) = d_{FD}(T_1, T_2)$.

Note: we technically only defined d_{FD} *for Reeb graphs. You can simply consider a merge tree to be the Reeb graph of itself (with regards to the same filter function).*

7.3 Mapper

Reeb graphs lose a lot of information, since they at most retain some 1-dimensional holes, but no larger holes. To generalize Reeb graphs further, we start looking at neighborhoods instead of levelsets, which will then lead to the Mapper algorithm.

7.3.1 An Approximation of the Reeb Graph

To begin, we consider the 1-dimensional case, and try to find an approximation of the Reeb graph. Instead of looking at pre-images of points, we will now look at pre-images of intervals. Let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ be an open cover of $\mathbb{R}$ (i.e., a collection of open sets whose union is $\mathbb{R}$). As always, we consider a function $f : X \to \mathbb{R}$. For each $f^{-1}(U_\alpha)$, we consider a partition into path-connected components, i.e., $f^{-1}(U_\alpha) = \bigcup_{\beta \in B_\alpha} V_\beta$. We then look at $f^*(\mathcal{U}) := \{V_\beta\}$, the set of all V_β we get over all α. Our object of interest is the nerve of this family, i.e., $N(f^*(\mathcal{U}))$.

If we take sufficiently nice functions, and sufficiently fine covers, then $N(f^*(\mathcal{U}))$ is isomorphic to R_f.

7.3.2 Topological Mapper

We can generalize this idea to maps to arbitrary spaces.

Definition 7.29. *Let* X, Z *be topological spaces. Then we call* $f : X \to Z$ well-behaved *if for all path-connected open sets* $U \subseteq Z$, $f^{-1}(U)$ *has finitely many path-connected components.*

Definition 7.30 (Mapper). *Let* $f : X \to Z$ *be well-behaved, and* $\mathcal{U}$ *be a (finite) open cover of* Z. *Then the* Mapper *is defined as* $M(\mathcal{U}, f) := N(f^*(\mathcal{U}))$.

As an example, we look at X being the boundary of the 3-cube $[0, 1]^3$. We then also look at $Z_1 = \mathbb{R}^2$ spanned by the x- and y-axis, with $f_1 : X \to Z_1$ being the projection onto this plane. Furthermore, we look at $Z_2 = \mathbb{R}$, spanned by just the x-axis, and $f_2 : X \to Z_2$ being again the projection.

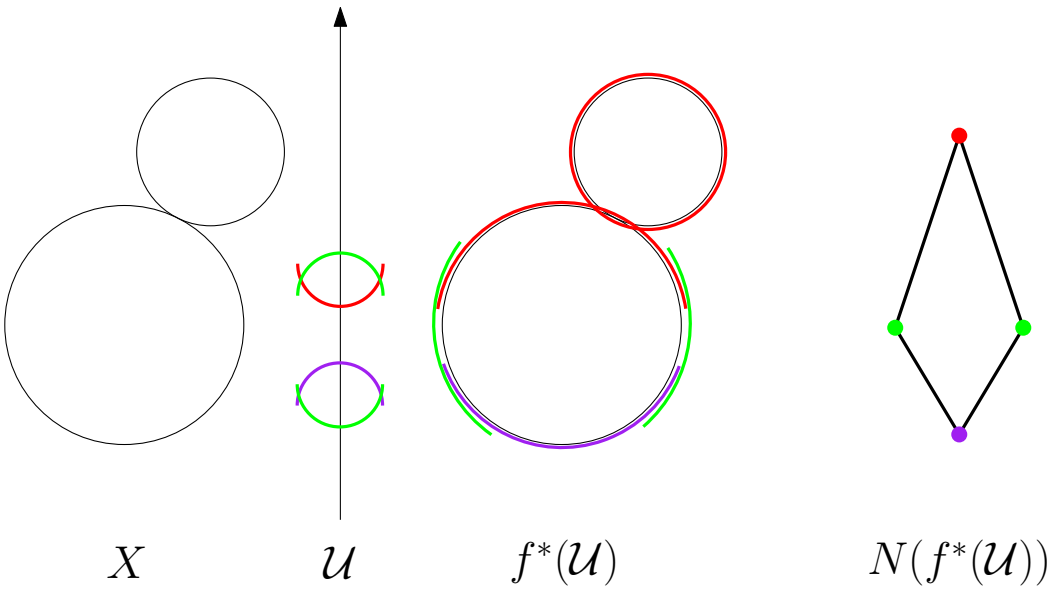

Figure 7.5: *A space* X*, an open cover* $\mathcal{U}$ *of* $\mathbb{R}$*, the family* $f^*(\mathcal{F})$*, and its nerve.*

We consider the open cover $\mathcal{U}_2$ of Z_2: $\{(-\infty, \frac{1}{3}), (0,1), (\frac{2}{3}, +\infty)\}$. For Z_1, we consider the cover $\mathcal{U}_1 := \mathcal{U}_2 \times \mathcal{U}_2$.

Exercise 7.31. *(a) Consider spaces* X, Z*, a filter function* $f : X \to Z$*, and an open cover* $\mathcal{U}$ *of* Z*. Show that if the* pullback cover $f^*(\mathcal{U})$ *is a* good cover *of* X*, then* $M(\mathcal{U}, f)$ *is homotopy equivalent to* X*.*

(b) Give an example of spaces X, Z*, a filter function* $f : X \to Z$*, and a* good cover $\mathcal{U}$ *of* Z*, such that* $M(\mathcal{U}, f)$ *is* not *homotopy equivalent to* X*.*

(c) Give an example of spaces X, Z*, a filter function* $f : X \to Z$*, and an open cover* $\mathcal{U}$ *of* Z *such that the pullback cover* $f^*(\mathcal{U})$ *is* not *a good cover, but* $M(\mathcal{U}, f)$ *is still homotopy equivalent to* X*.*

7.3.3 Mapper for Point Clouds

We would like to apply the ideas of the topological Mapper to analyze the shape of data. However, once again we have the issue that data usually does not come in the form of a topological space, but as a set of data points with a notion of distance between them. The *Mapper algorithm* for point clouds adapts the ideas of the topological Mapper to this setting.

Input: In the most general setting, data comes as a finite metric space (P, d_P), for example as points in $\mathbb{R}^d$ or as vertices of a graph. We also require a cover $\mathcal{U}$ of a space Z, usually $Z = \mathbb{R}$, as input. Finally, we also need a filter function $f : P \to Z$ and a clustering algorithm (which might also require some input parameters).

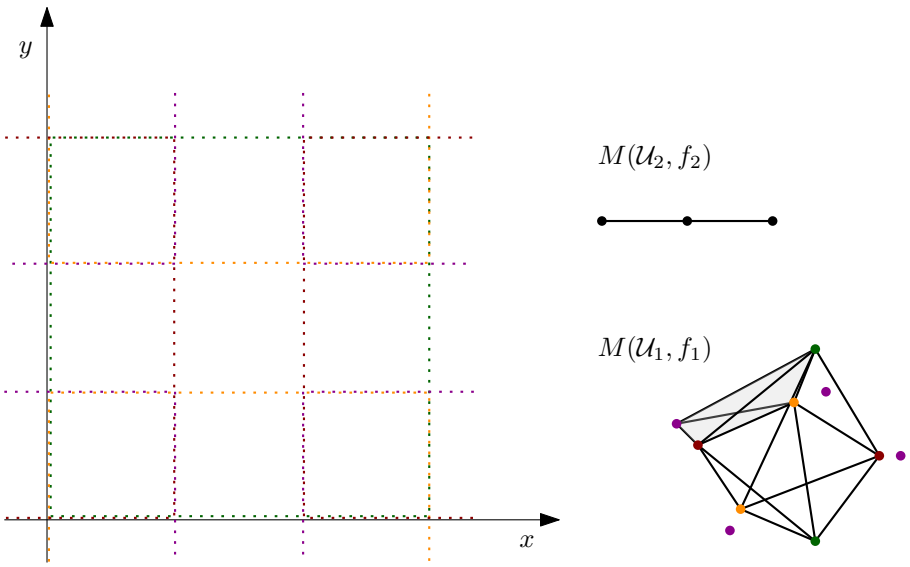

Figure 7.6: *The cover $\mathcal{U}_\infty$, and the two Mappers. The Mapper $M(\mathcal{U}_1, f_1)$ consists of an empty octahedron, with additional filled tetrahedra attached at the purple vertices. The whole space thus collapses to an octahedron.*

Algorithm: Since at the moment we only have a discrete metric space, we do not really have the notion of connected components yet. For every $U \in \mathcal{U}$, we thus cluster the pre-image $f^{-1}(U)$ using some clustering algorithm, which we can also consider as an input. Now, we can just consider each cluster C_i as a vertex of some simplicial complex K, and add a face $\{C_1, \dots, C_k\}$ to K if these clusters (which are just point sets) have a common point.

Output: We output K, or even just its 1-skeleton.

As you can see, this algorithm requires a lot of input parameters. While this allows to encode previous knowledge of the data set (e.g. by choosing as filter function the distance to a known center of the data), it also makes the space of possible outputs very large. Picking the correct parameters is currently still an art form on its own, and there is significant research being done towards understanding the interplay between the parameters and statistical guarantees for certain good choices of parameters.

7.4 Multiscale Mapper

Motivated by the many tuneable parameters, we discuss here one idea to look at many values at once. The multiscale Mapper is a combination of the ideas of persistence and of Mapper. We here want to look at different covers.

Definition 7.32. *Let $\mathcal{U} = \{U_\alpha\}_{\alpha \in A}$ and $\mathcal{V} = \{V_\beta\}_{\beta \in B}$ be two covers of the same space X. A* map of covers *is a map $\varphi : A \to B$ such that for every $\alpha \in A$, we have $U_\alpha \subseteq V_{\varphi(\alpha)}$.*

Proposition 7.33. *If $\varphi : \mathcal{U} \to \mathcal{V}$ is a map of covers (with a slight abuse of notation), then the map $N(\varphi) : N(\mathcal{U}) \to N(\mathcal{V})$ given on the vertices by φ is simplicial.*

Proof. Let $\sigma \in N(\mathcal{U})$. We need to show that the intersection $\bigcap_{\beta \in N(\varphi)(\sigma)} V_\beta$ is non-empty.

$$\bigcap_{\beta \in N(\varphi)(\sigma)} V_\beta = \bigcap_{\alpha \in \sigma} V_{\varphi(\alpha)} \supseteq \bigcap_{\alpha \in \sigma} U_\alpha \neq \emptyset$$

Thus, $N(\varphi)(\sigma) \in N(\mathcal{V})$. □

An example of this map between nerves can be seen in Figure 7.7.

Proposition 7.34. *Let $f : X \to Z$ be some map, and $\mathcal{U}, \mathcal{V}$ be covers of Z, with $\varphi : \mathcal{U} \to \mathcal{V}$ some map of covers. Then, there exists a map of covers $f^*(\varphi) : f^*(\mathcal{U}) \to f^*(\mathcal{V})$.*

Recall that $f^*(\mathcal{U})$ is the cover of X consisting of the connected components of the pre-images of the sets of $\mathcal{U}$ under f.

Proof. For every α, we have $U_\alpha \subseteq V_{\varphi(\alpha)} \implies f^{-1}(U_\alpha) \subseteq f^{-1}(V_{\varphi(\alpha)})$. We now need to go from these pre-images to their connected components. Since every connected component of $f^{-1}(U_\alpha)$ must lie in a unique connected component of $f^{-1}(V_{\varphi(\alpha)})$, our desired map of covers is given by exactly mapping to this connected component. □

If we have multiple maps of covers, $\mathcal{U} \xrightarrow{\varphi} \mathcal{V} \xrightarrow{\psi} \mathcal{W}$, we can concatenate the maps, and the f^* function distributes: $f^*(\psi \circ \varphi) = f^*(\psi) \circ f^*(\varphi)$.

Let $\mathfrak{U} = \mathcal{U}_1 \xrightarrow{\varphi_1} \mathcal{U}_2 \xrightarrow{\varphi_2} \dots \xrightarrow{\varphi_{n-1}} \mathcal{U}_n$ be a sequence of covers of Z with maps between them, which we call a *cover tower*. By applying f^* we get a cover tower $f^*(\mathfrak{U})$ of X.

Definition 7.35 (Multiscale Mapper). *Let $f : X \to Z$, $\mathfrak{U}$ a cover tower of Z. Then, the* Multiscale Mapper $MM(\mathfrak{U}, f)$ *is*

$$MM(\mathfrak{U}, f) := N(f^*(\mathfrak{U})) = \{N(f^*(\mathcal{U}_i)) \mid \mathcal{U}_i \in \mathfrak{U}\}$$

together with the induced simplicial maps

$$N(f^*(\varphi_i)) : N(f^*(\mathcal{U}_i)) \to N(f^*(\mathcal{U}_{i+1})).$$

Applying homology, we get the sequence homology groups with induced homomorphisms between them, i.e., a persistence module:

$$H_p(N(f^*(\mathcal{U}_1))) \xrightarrow{N(f^*(\varphi_1))_*} \dots \xrightarrow{N(f^*(\varphi_{n-1}))_*} H_p(N(f^*(\mathcal{U}_n))).$$

We can now view $Dgm_p MM(\mathfrak{U}, f)$ as a topological summary of f through the lens of $\mathfrak{U}$.

As opposed to the normal Mapper, at first glance the Multiscale Mapper adds even more parameters. But a cover tower can be seen as a way of looking at a whole interval of covers. For example, we can get a cover tower by increasing the size of all intervals in an interval cover. The features of the data should show up as a robust feature that persists for a longer time over this process, while spurious features obtained from choosing "wrong" Mapper parameters should disappear quickly.

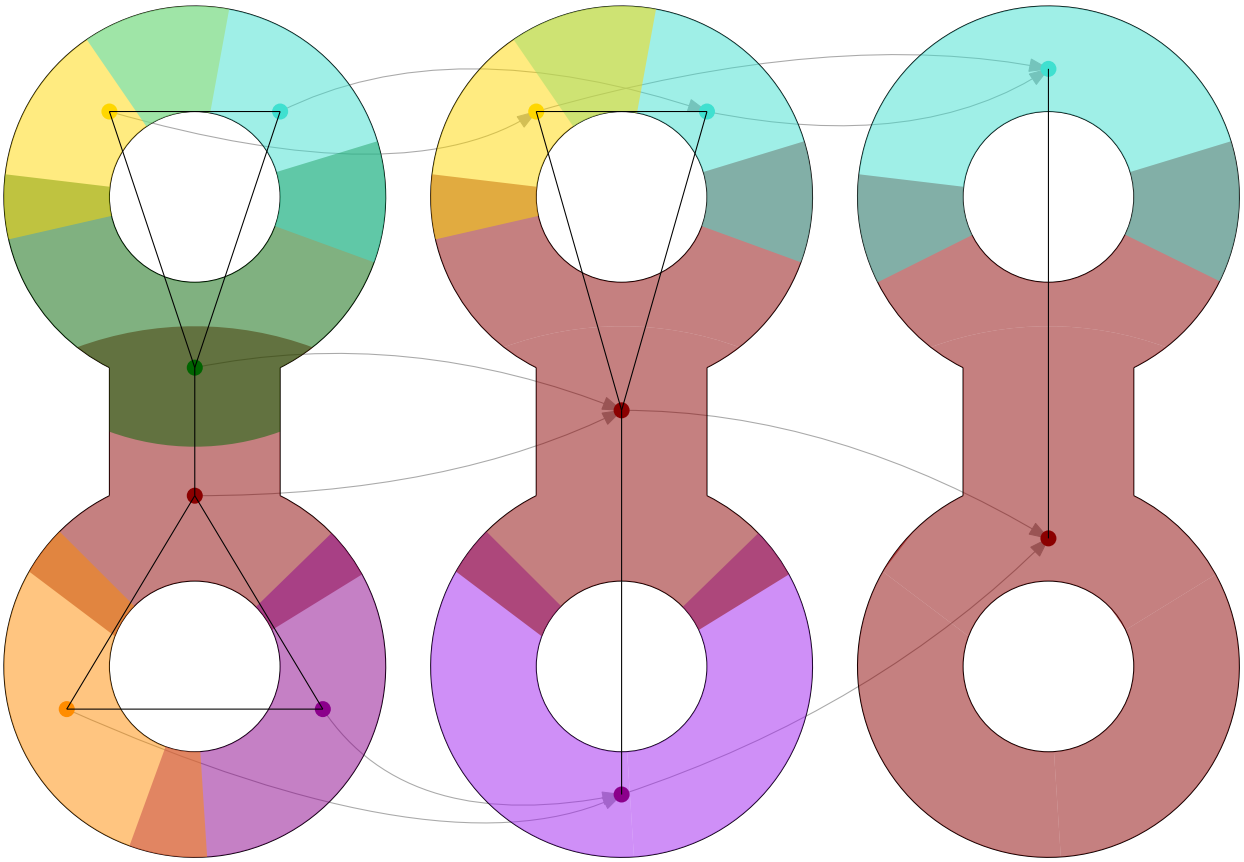

Figure 7.7: *Three pullback covers of a space X (thickened figure "8"), with their nerves and the induced simplicial maps between them.*

Questions

24. *What is a Reeb graph?* State the definition and describe how we get the graph structure.
25. *How can we compute the augmented Reeb graph of a piece-wise linear function?* Define the augmented Reeb graph and explain the algorithm to compute it.
26. *How much of the homology of the underlying topological space is captured by the Reeb graph?* Explain vertical and horizontal homology.
27. *What is the interleaving distance for Reeb graphs?* Give the definitions and state the relevant stability theorems.
28. **(This topic was not covered in this year's course in FS25 and therefore the following question will not be asked in the exam.)** *What is the functional distortion distance for Reeb graphs?* Give the definitions and state the relevant stability theorems.
29. *What is the topological Mapper?* State the Definition and give an example.
30. *How can we use Mapper on point cloud data?* Explain the Mapper algorithm and describe the input parameters.
31. **(This topic was not covered in this year's course in FS25 and therefore the following question will not be asked in the exam.)** *How can we use Mapper on several covers at once?* Explain the Multiscale Mapper.