Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Patrick Schnider

Introduction to Topological Data Analysis — Scribe Notes 21 — FS23

Scribe notes by Simon Weber. Please contact me for corrections.
Lecture date: May 12, 2023
Last update: Friday 19th May, 2023, 07:40

Definition 1. *Let* R_f *be a Reeb graph of a space* X*, and* $u, v \in R_f$ *(in the same connected component), 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 2 (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:*

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

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

And finally, we define the functional distortion distance*,*

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

Theorem 3. *Let* $f, g : X \to \mathbb{R}$ *be tame functions. Then,* $d_{FD}(R_f, R_g) \leq ||f - g||_\infty$*.*

Theorem 4. $d_I(R_f, R_g) \leq d_{FD}(R_f, R_g) \leq 3d_I(R_f, R_g)$*.*

Mapper

An approximation of the Reeb graph

Reeb graphs "throw away" 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.

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., $N(f^*(\mathcal{U}))$.

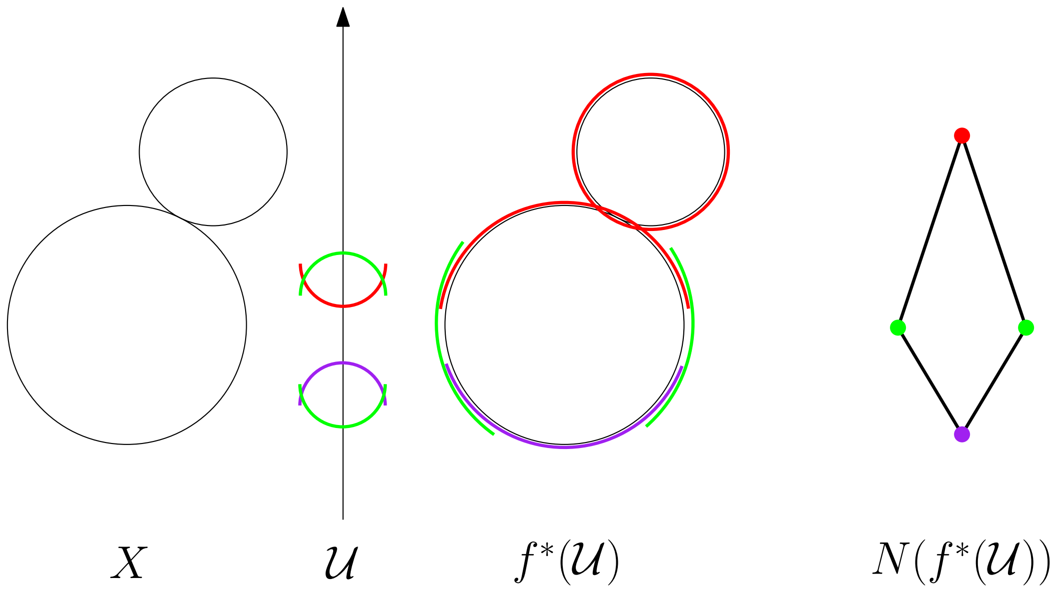

Figure 1: A space X, an open cover $\mathcal{U}$ of $\mathbb{R}$, the family $f^*(\mathcal{F})$, and its nerve.

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

Topological Mapper

Definition 5. *Let X, Z be topological spaces. Then we call $f : X \to Z$* well-behaved *if for all open sets $U \subseteq Z$, $f^{-1}(U)$ has finitely many path-connected components.*

Definition 6 (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 $M(\mathcal{U}, f) := N(f^*(\mathcal{U}))$.*

Example: 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.

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 $\mathcal{U}_1 := \mathcal{U}_2 \times \mathcal{U}_2$.

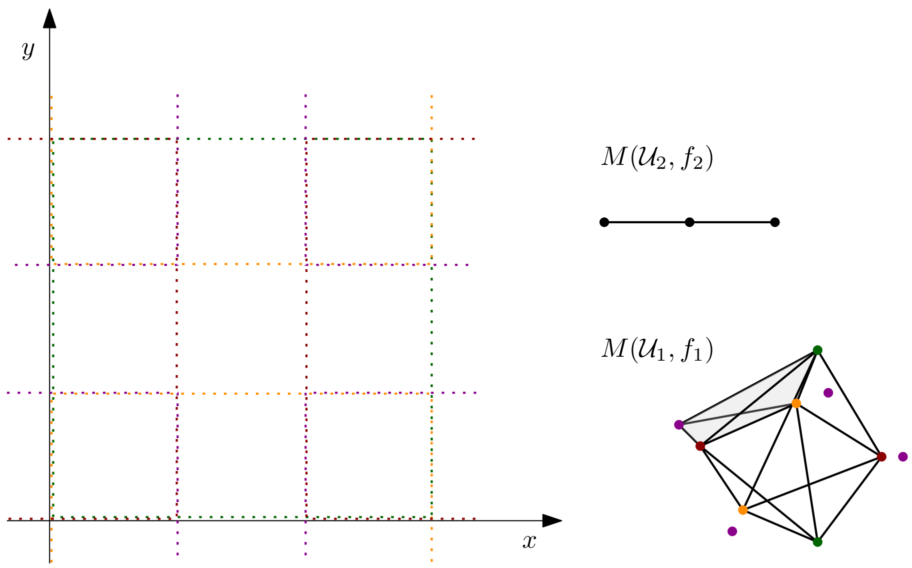

Figure 2: The cover $\mathcal{U}_\infty$, and the two Mappers. The Mapper $M(\mathcal{U}_1, f_1)$ consists of an empty octahedron, with additional filled tetrahedra attached at the purple vertices. The whole space thus collapses to an octahedron.

Mapper for Point Clouds

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 requires a cover $\mathcal{U}$ of a space Z, usually $Z = \mathbb{R}$, as input. Finally, we also need a filter function $f : P \to Z$.

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 preimage $f^{-1}(\mathcal{U})$ 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 $\{C_1, \ldots, C_k\}$ to K if these clusters (which are just point sets) have a common point.

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