Differentiable Mapper for Topological Optimization of Data Representation

Ziyad Oulhaj [1] & Mathieu Carrière [2] & Bertrand Michel [3]

[1] *Nantes Université, Centrale Nantes, Laboratoire de Mathématiques Jean Leray, France, ziyad.oulhaj@ec-nantes.fr*
[2] *DataShape, Centre Inria d'Université Côte d'Azur, France, mathieu.carriere@inria.fr*
[3] *Nantes Université, Centrale Nantes, Laboratoire de Mathématiques Jean Leray, France, bertrand.michel@ec-nantes.fr*

Résumé. La représentation et la visualisation non supervisées de données à l'aide d'outils de topologie constituent un domaine actif et en expansion de l'Analyse de Données Topologiques (TDA) et de la science des données. Une de ses lignes de travail les plus remarquables repose sur le graphe Mapper, qui est un graphe combinatoire dont les structures topologiques (composantes connexes, branches, boucles) correspondent à celles des données elles-mêmes. Bien que très générique et applicable, son utilisation a été jusqu'à présent entravée par l'ajustement manuel de ses nombreux paramètres, parmi lesquels un crucial est le filtre : il s'agit d'une fonction continue dont les variations sur l'ensemble de données sont l'ingrédient principal à la fois pour construire la représentation du Mapper et pour évaluer la présence et les tailles de ses structures topologiques. Cependant, il n'existe actuellement aucune méthode pour ajuster le filtre lui-même. Dans ce travail, nous nous appuyons sur un cadre d'optimisation récemment proposé incorporant la topologie pour fournir le premier schéma d'optimisation du filtre pour les graphes Mapper. Pour y parvenir, nous proposons une version plus relaxée et plus générale du graphe Mapper, dont les propriétés de convergence sont étudiées.

Mots-clés. Graphe Mapper, Visualisation de Données, Analyse de Données Topologiques, Homologie Persistante.

Abstract. Unsupervised data representation and visualization using tools from topology is an active and growing field of Topological Data Analysis (TDA) and data science. Its most prominent line of work is based on the so-called *Mapper graph*, which is a combinatorial graph whose topological structures (connected components, branches, loops) are in correspondence with those of the data itself. While highly generic and applicable, its use has been hampered so far by the manual tuning of its many parameters—among these, a crucial one is the so-called *filter*: it is a continuous function whose variations on the data set are the main ingredient for building the Mapper representation. However, there is currently no method for tuning the filter itself. In this work, we build on a recently proposed optimization framework incorporating topology to provide the first filter optimization scheme for Mapper graphs. In order to achieve this, we propose a relaxed and more general version of the Mapper graph, whose convergence properties are investigated.

Keywords. Mapper Graph, Data Visualization, Topological Data Analysis, Persistent Homology.

1 Introduction

Mapper graphs and TDA. The Mapper graph introduced in [1] is an essential tool of Topological Data Analysis (TDA), and has been used many times for visualization purposes on different types of data, including, but not limited to, single-cell sequencing [2, 3], neural network architectures [4, 5], or 3D meshes [6, 7]. Moreover, its ability to precisely encode (within the graph) the presence and sizes of geometric and topological structures in the data in a mathematically founded way (through the use of algebraic topology) has also proved beneficial for highlighting subpopulations of interest, which are usually detected as peculiar topological structures of significant sizes, and identifying the key features that best explain such subpopulations against the rest of the Mapper graph. This general pipeline has become a key component in, e.g., biological inference in single-cell data sets, where differentiating stem cells can usually be recovered from branching patterns in the corresponding Mapper graphs [8].

Contributions. Our contribution is three-fold:

- We introduce *Soft Mapper*: a generalization of the combinatorial Mapper graph in the form of a probability distribution on Mapper graphs,
- We propose a filter optimization framework adapted to a smooth Soft Mapper distribution with provable convergence guarantees,
- We implement and showcase the efficiency of Mapper filter optimization through Soft Mapper on various data sets, with public, open-source code in `TensorFlow`.

2 Background on Reeb and Mapper graphs

Reeb graphs. Mapper graphs can be understood as numerical approximations of *Reeb graphs*, that we now define. Let X be a topological space and let $f : X \to \mathbb{R}$ be a continuous function called *filter function*. Let $\sim_f$ be the equivalence relation between two elements x and y in X defined by: $x \sim_f y$ if and only if x and y are in the same connected component of $f^{-1}(z)$ for some z in $f(X)$. The Reeb graph $\mathrm{R}_f(X)$ of X is then simply defined as the quotient space $X/\sim_f$.

Mapper graphs. The Mapper was introduced in [1] as a discrete and computable version of the Reeb graph $\mathrm{R}_f(\mathcal{X})$. Assume that we are given a point cloud $\mathbb{X}_n = \{X_1, \ldots, X_n\} \subseteq \mathcal{X}$ with known pairwise dissimilarities, as well as a filter function f computed on each point of $\mathbb{X}_n$. The Mapper graph can then be computed with the following generic version of the Mapper algorithm:

1. Cover the range of values $\mathbb{Y}_n = f(\mathbb{X}_n)$ with a set of consecutive intervals $I_1, \ldots, I_r$ that overlap, i.e., one has $I_i \cap I_{i+1} \neq \varnothing$ for all $1 \leq i \leq r-1$.

2. Apply a clustering algorithm to each pre-image $f^{-1}(I_j)$, $j \in \{1, ..., r\}$. This defines a *pullback cover* $\mathcal{C} = \{\mathcal{C}_{1,1}, \ldots, \mathcal{C}_{1,k_1}, \ldots, \mathcal{C}_{r,1}, \ldots, \mathcal{C}_{r,k_r}\}$ of $\mathbb{X}_n$.

3. The Mapper graph is defined as the *nerve* of $\mathcal{C}$. Each node $v_{j,k}$ of the Mapper graph corresponds to an element $\mathcal{C}_{j,k}$ of $\mathcal{C}$, and two nodes $v_{j,k}$ and $v_{j',k'}$ are connected by an edge if and only if $\mathcal{C}_{j,k} \cap \mathcal{C}_{j',k'} \neq \varnothing$.

3 Soft Mapper construction

In this section, we introduce our new construction *Soft Mapper*, which generalizes Mapper graphs and can be used for non-convex optimization. In order to do so, we first provide a general formalization of the Mapper construction that does *not* require overlapping intervals and filter functions. Then, we use this formalization to define *Soft Mapper*, which essentially consists in a distribution on regular Mapper graphs.

3.1 Mapper graphs built on latent cover assignments

Let $\mathbb{X}_n = \{x_1, ..., x_n\}$ be a point cloud lying in a metric space (X, d) and let $r \in \mathbb{N}^\star$. For instance, $\mathbb{X}_n$ can be obtained from sampling X^n according to some distribution μ. Then, let Clus be a clustering algorithm on (X, d), that is assumed to be fixed in the following of this work.

Latent cover assignments. Any binary matrix $e \in \{0, 1\}^{n \times r}$ is then called an *r-latent cover assignment* of $\mathbb{X}_n$, where $e_{i,j} = 1$ must be understood as point x_i belonging to the j-th element of a *latent cover* of the data. For instance, in the standard version of Mapper presented in Section 2, the latent cover is obtained from a family of pre-images of intervals that cover the range of the filter function.

The procedure to compute a Mapper graph given an r-latent cover assignment $e \in \{0, 1\}^{n \times r}$ is straightforward: simply replace $f^{-1}(I_j)$ by $\{x_i : e_{i,j} = 1\}$ in the generic Mapper algorithm in Section 2, then derive the pullback cover using the clustering algorithm Clus, and finally compute the Mapper graph as the nerve of the pullback cover.

Mapper function. Let $\mathbb{K}$ be the set of simplicial complexes of dimension less or equal to 1 (i.e., graphs) and such that their sets of vertices (i.e., their $0-$skeletons) are subsets of the power set $\mathcal{P}(\mathbb{X}_n)$. We define the Mapper complex generating function as:

$$\text{MapComp}\colon \{0, 1\}^{n \times r} \longrightarrow \mathbb{K},$$

where MapComp takes a latent cover assignment as input and creates the corresponding Mapper graph.

3.2 Cover assignment scheme and Soft Mapper

Now, we define stochastic schemes for generating latent cover assignments, that we call *cover assignment schemes*.

Definition 3.1. A *cover assignment scheme* is a double indexed sequence of random variables

$$A = (A_{i,j})_{\substack{1\leq i\leq n\\ 1\leq j\leq r}}$$

such that each $A_{i,j}$ is a Bernoulli random variable conditionally to $\mathbb{X}_n$. Let $p_{i,j}(\mathbb{X}_n)$ be the parameter of the Bernoulli distribution of $(A_{i,j}|\mathbb{X}_n)$, which is thus a function of the point cloud $\mathbb{X}_n$.

Definition 3.2. Let A be a cover assignment scheme. The *Soft Mapper* of A is defined as the associated distribution of Mapper complexes, which corresponds to the push forward measure of the distribution of A by the map MapComp.

4 Smooth relaxation of the standard cover assignment scheme

Given some $\delta > 0$, we can now define a cover assignment scheme A_δ that approximates the degenerate cover assignment scheme A^* arising from the standard Mapper graph. Specifically, using the same notations as before, and denoting each element of the cover with $I_j = [a_j, b_j]$, consider, for each $j \in \{1, ..., r\}$, the function $q_j \colon X \longrightarrow [0, 1]$ defined with:

$$x \mapsto \begin{cases} 1, & \text{if } f(x) \in [a_j, b_j] \\ \exp(1 - 1/(1 - (\frac{a_i - f(x)}{\delta})^2)), & \text{if } f(x) \in [a_j - \delta, a_j] \\ \exp(1 - 1/(1 - (\frac{f(x) - b_i}{\delta})^2)), & \text{if } f(x) \in [b_j, b_j + \delta] \\ 0, & \text{otherwise} \end{cases}$$

Now, define $A_\delta = (A_{\delta,i,j})_{\substack{1\leq i\leq n\\ 1\leq j\leq r}}$ to be the random variable in $\{0, 1\}^{n\times r}$ such that for every $(i, j) \in \{1, ..., n\} \times \{1, ..., r\}$:

$$A_{\delta,i,j} \mid \mathbb{X}_n \sim \mathcal{B}(q_j(x_i)),$$

with the $A_{\delta,i,j}$'s being jointly independent conditionally to $\mathbb{X}_n$. As for the standard cover, the Bernoulli parameter $p_{i,j} = q_j(x_i)$ depends on its associated point x_i and also on the chosen filter f.

Moreover, notice that for every $x_i \in \mathbb{X}_n$ and $j \in \{1, ..., r\}$:

$$q_j(x_i) \xrightarrow[\delta\to 0]{} \begin{cases} 1, & \text{if } f(x_i) \in I_j \\ 0, & \text{otherwise,} \end{cases}$$

and this shows that $A_\delta \xrightarrow[\delta\to 0]{\mathcal{L}} A^*$.

5 Topological risk of Soft Mappers

We now switch to the problem of designing filter functions automatically for Mapper graphs using Soft Mapper. To answer this ill-posed problem, we propose to look for filter functions that are optimal with respect to some topological criteria associated to their (Soft)Mapper graphs. In particular, we focus on topological losses based on *persistent homology*.

5.1 Topological signature for Mapper graphs

Persistent homology. Persistent homology is a powerful tool that allows to encode the topological information contained in a nested family of simplicial complexes, also called a *filtered simplicial complex*, see for instance [9] for a general introduction. In the context of Mapper graphs, a variation of persistent homology called *extended* persistent homology has been proved useful, as applying it on Mapper graphs produces descriptors called *extended persistence diagrams*. These diagrams only require to define a *filtration function* on the graph, and are made of points in the Euclidean plane, each point encoding the presence and size (w.r.t. the filtration function) of a particular topological structure of the Mapper graph (such as a connected component, a branch or a loop).

We now define a filtration function on Mapper graphs in order to compute extended persistence diagrams. Let $\mathcal{F}(\mathbb{X}_n, \mathbb{R})$ be the space of real valued functions defined on the point cloud $\mathbb{X}_n$. For a function $F \in \mathcal{F}(\mathbb{X}_n, \mathbb{R})$, we first associate a filtration ϕ to some $K \in \mathrm{im}(\mathrm{MapComp})$ with:

$$\forall \sigma \in K \ : \ \phi(\sigma) = \max_{c \in \sigma} \frac{\sum_{x \in c} F(x)}{\mathrm{card}(c)},$$

that is, node filtration values are defined as the average filter values of the data points associated to the node, and edge filtration values are computed as the maximum of their node values. Then, we compute the extended persistence diagram (which we consider as a subset of $\mathbb{R}^2$) of the filtered simplicial complex (K, ϕ). We denote by MapPers the function that takes a Mapper graph and a scalar function on $\mathbb{X}_n$, and then outputs the persistence diagram:

$$\mathrm{MapPers}\colon \mathbb{K} \times \mathcal{F}(\mathbb{X}_n, \mathbb{R}) \longrightarrow \mathcal{P}(\mathbb{R}^2).$$

Persistence specific loss. Now, we introduce a generic notation for a loss function—or, more simply, a statistic—that associates a real value to any extended persistence diagram. Denoting PD as the set of subsets of $\mathbb{R}^2$ consisting of a finite number of points outside the diagonal $\Delta = \{(x, x) : x \in \mathbb{R}\}$, such a function can be written as $\ell\colon PD \longrightarrow \mathbb{R}$.

5.2 Statistical risk of the topological signature associated to Soft Mapper

We finally compute the loss associated to a Mapper graph with the function

$$\begin{aligned}\mathcal{L}\colon \{0,1\}^{n\times r} \times \mathcal{F}(\mathbb{X}_n, \mathbb{R}) &\longrightarrow \mathbb{R} \\ (e, F) &\longmapsto \ell\left(\mathrm{MapPers}\left(\mathrm{MapComp}(e), F\right)\right).\end{aligned}$$

Then, we define the risk of a Soft Mapper $\mathrm{MapComp}(A)$ by integrating the loss according to the distribution of the Soft Mapper, or equivalently according to the distribution of the cover assignment scheme:

$$\mathbb{E}\left(\mathcal{L}(A, F)|\mathbb{X}_n\right) = \sum_{e\in\{0,1\}^{n\times r}} \mathcal{L}(e, F) \cdot \mathbb{P}(A = e|\mathbb{X}_n).$$

6 Conditional risk optimization with respect to parameters

Now that we have properly defined risks associated to Soft Mapper distributions, we study in this section the convergence properties of filter optimization schemes minimizing such risks.

6.1 Problem setting

Let us introduce a parameterized family of functions $\{f_\theta : \mathbb{X}_n \to \mathbb{R},\, \theta \in \mathbb{R}^s\}$. In order to simplify notations, we assume in the following of the article that the function F used to compute persistence diagrams and the filter function f_θ used to design cover assignments are the same, $F = f_\theta$. Let A be a cover assignment scheme whose joint distribution $\mathbb{P}_\theta$ depends on the filter function f_θ.

Our goal is to find the optimal set of parameters $\bar{\theta}$ that minimizes the topological risk associated to $\mathrm{MapComp}(A)$, when f_θ is used to define the filtration values on the Mapper graphs. In other words, if we denote:

$$\begin{aligned}\mathrm{L}\colon \mathbb{R}^s &\longrightarrow \mathbb{R} \\ \theta &\longmapsto \mathbb{E}_\theta(\mathcal{L}(A, f_\theta)|\mathbb{X}_n),\end{aligned} \tag{1}$$

our aim is to find a minimizer of L. Note that in the definition of L, the expectation depends on θ because the distribution of A also depends on it.

In order to prove guarantees about minimizing L, we follow [10], which uses the theoretical background introduced in [11], in which the authors prove that stochastic gradient descent algorithms converge under certain conditions. To use this framework, it suffices to prove two points (see Corollary 5.9. in [11]):

- L is definable in an o-minimal structure (see [12]),
- L is locally Lipschitz.

6.2 Theoretical guarantees on the convergence of a gradient descent scheme

Under regularity assumptions on the parameterized family of filter functions $\mathcal{F} = \{f_\theta \colon \mathbb{X}_n \longrightarrow \mathbb{R},\, \theta \in \mathbb{R}^s\}$, we now show that the risk L in Equation (1) is definable and smooth.

Theorem 6.1. *Suppose that there exists an o-minimal structure $\mathcal{S}$ such that:*

- *for every $x \in \mathbb{X}_n$, the function $\theta \mapsto f_\theta(x)$ is definable in $\mathcal{S}$ and is locally Lipschitz,*
- *for every $m \in \mathbb{N}$, the restriction of ℓ to the set of (extended) persistence diagrams of size m is definable in $\mathcal{S}$ and is locally Lipschitz,*
- *for every $e \in \{0,1\}^{n \times r}$, the function $\theta \mapsto \mathbb{P}_\theta(A = e|\mathbb{X}_n)$ is definable in $\mathcal{S}$ and is locally Lipschitz.*

Then L *is definable in $\mathcal{S}$ and is locally Lipschitz.*

Under the assumptions of Theorem 6.1, it is then possible to give guarantees on the convergence of a stochastic gradient descent scheme to some critical points of L. This only requires additional, but mild and not very restrictive technical conditions regarding the stochastic gradient descent algorithm itself.

7 Numerical Experiments

A first application where we can use the Soft Mapper optimization setting is finding linear filters in order to skeletonize 3-dimensional shapes with Mapper graphs. Here, our dataset $\mathbb{X}_n$ consists each time of a point cloud embedded in $\mathbb{R}^3$. The different point clouds we study are displayed (as meshes) in Figure 1. The parametric family of functions is linear, i.e., equal to $\{f_\theta \colon x \mapsto \langle x, \theta \rangle,\, \theta \in \mathbb{R}^3\}$, and the cover assignment scheme A_δ is the smooth relaxation of the standard case, with $\delta = 10^{-2} \cdot (\max_{x \in \mathbb{X}_n} f_\theta(x) - \min_{x \in \mathbb{X}_n} f_\theta(x))$. The cover of the image space is given by r intervals of the same length, such that consecutive intervals have a percentage g of their length in common. The clustering algorithm for the three shapes is KMeans.

Objective. Intuitively, the optimal directions to filter the 3-dimensional shapes (in a topological sense) are:

- for the human: the vertical direction,
- for the octopus: the parallel direction to its tentacles,
- for the table: the perpendicular direction to its upper surface,

as these directions induce Mapper graphs with more topological structures. We will therefore measure the quality of our results by comparing our optimized directions $\bar{\theta}$ to the ones cited above. To find $\bar{\theta}$, we use the total (regular) persistence as a persistence specific loss ℓ, each time taking the diagonal as the initial direction, i.e. $\theta_0 = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^T$. The learning curves for each 3-dimensional shape are displayed in Figure 2, and the correlations between the optimized directions and those we identified as intuitively optimal are summarized in Table 1. As one can see from the table, we are able to recover these intuitive directions with gradient descent.

Qualitative assessment. One can see, in Figures 3 and 4, that the regular Mapper graphs built with the initial and final (optimized) filter functions show clear improvement in the ability of the graphs to act as skeletons of the original point clouds. The third shape, representing a table, is particularly interesting. Indeed, the optimal direction that we captured is different from the first and the second principal components computed by PCA, since the principal plane of the point cloud is given by the surface of the table.

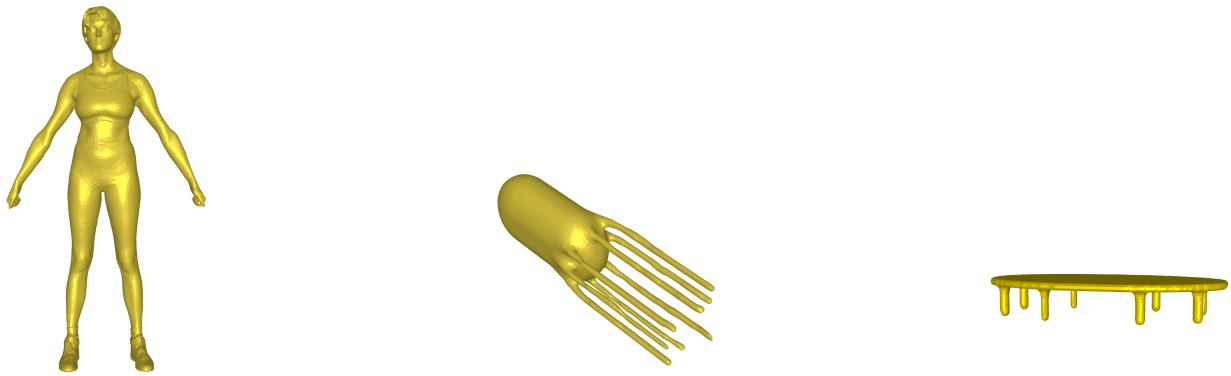

Figure 1: Meshes of 3-dimensional point clouds representing from left to right: a human, an octopus and a table.

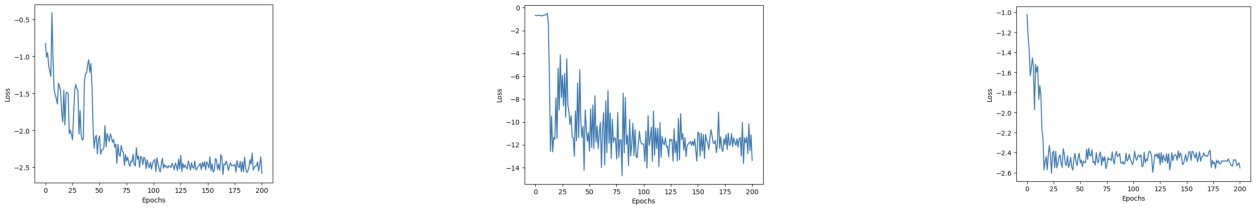

Figure 2: Learning curves for the 3-dimensional shapes corresponding, from left to right, to: the human, the octopus and the table.

Human	Octopus	Table
0.9999	-0.9984	0.9993

Table 1: Correlation between the optimized directions and the optimal ones, for each 3-dimensional shape.

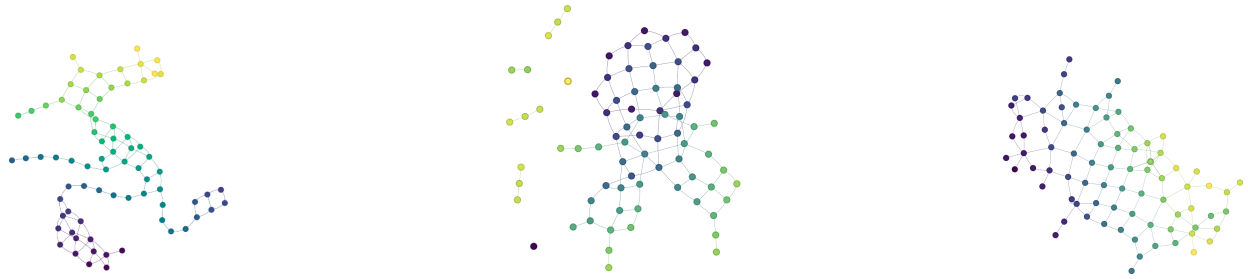

Figure 3: Regular Mapper graphs computed with the initial filter function, corresponding, from left to right, to: the human, the octopus and the table. Vertices are colored using the mean value of the filter function in the corresponding clusters.

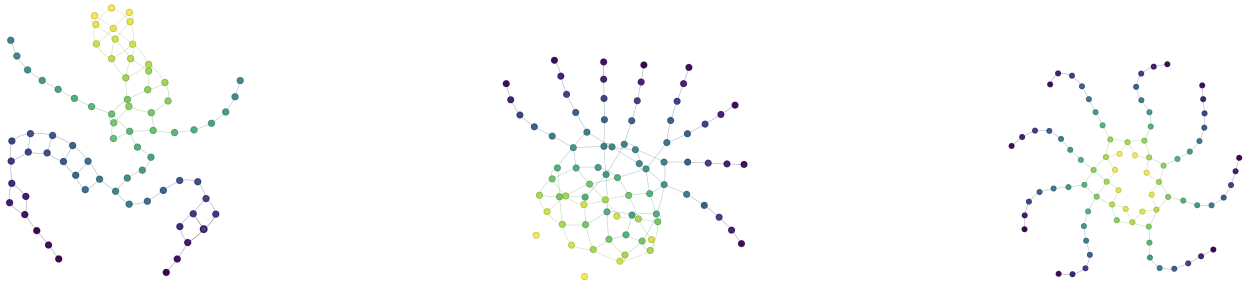

Figure 4: Regular Mapper graphs computed with the optimized filter function, corresponding, from left to right, to: the human, the octopus and the table. Vertices are colored using the mean value of the filter function in the corresponding clusters.

8 Discussion

In this article, we have introduced Soft Mapper, a distributional and smoother version of the standard Mapper graph, with provable convergence guarantees using persistence-based losses and risks. Our case study in this article was finding an optimal filter function, among a parameterized family of functions, in order to construct regular Mapper graphs incorporating an optimized and maximal amount of topological information. We then produced examples of such optimization processes on real 3D shape data, for which we were able to obtain structured Mapper representations in an unsupervised way.

Bibliographie

References

[1] Gurjeet Singh, Facundo Mémoli, Gunnar E Carlsson, et al. Topological methods for the analysis of high dimensional data sets and 3d object recognition. *PBG@ Eurographics*, 2:091–100, 2007.

[2] Tongxin Wang, Travis Johnson, Jie Zhang, and Kun Huang. Topological methods for visualization and analysis of high dimensional single-cell rna sequencing data. In *BIOCOMPUTING 2019: Proceedings of the Pacific Symposium*, pages 350–361. World Scientific, 2018.

[3] Sabrina Zechel, Pawel Zajac, Peter Lönnerberg, Carlos F Ibáñez, and Sten Linnarsson. Topographical transcriptome mapping of the mouse medial ganglionic eminence by spatially resolved rna-seq. *Genome biology*, 15:1–12, 2014.

[4] Satanik Mitra and Kameshwar Rao JV. Experiments on fraud detection use case with qml and tda mapper. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 471–472, 2021.

[5] Brian B Joseph, Trami Pham, and Christopher Hastings. Topological data analysis in conjunction with traditional machine learning techniques to predict future mdap pm ratings. Acquisition Research Program, 2021.

[6] Ziqi Wang. Exploration of topological data analysis in 3d printing. In *2020 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pages 150–153. IEEE, 2020.

[7] Paul Rosen, Mustafa Hajij, Junyi Tu, Tanvirul Arafin, and Les Piegl. Inferring quality in point cloud-based 3d printed objects using topological data analysis. *arXiv preprint arXiv:1807.02921*, 2018.

[8] Abbas Rizvi, Pablo Cámara, Elena Kandror, Thomas Roberts, Ira Schieren, Tom Maniatis, and Raúl Rabadán. Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development. *Nature Biotechnology*, 35:551–560, 2017.

[9] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Society, 2010.

[10] Mathieu Carriere, Frédéric Chazal, Marc Glisse, Yuichi Ike, Hariprasad Kannan, and Yuhei Umeda. Optimizing persistent homology based functions. In *International conference on machine learning*, pages 1294–1303. PMLR, 2021.

[11] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020.

[12] Michel Coste. *An introduction to o-minimal geometry*. Istituti editoriali e poligrafici internazionali Pisa, 2000.