ATOL: Automatic Topologically-Oriented Learning

Martin Royer, Frédéric Chazal, Yuichi Ike, Yuhei Umeda

ATOL: Automatic Topologically-Oriented Learning

Martin Royer and **Frederic Chazal**
Datashape, Inria Saclay
Palaiseau, France.
firstname.surname@inria.fr

Yuichi Ike and **Yuhei Umeda**
Fujitsu Laboratories, AI Lab
Tokyo, Japan.
surname.firstname@fujitsu.com

Abstract

There are abundant cases for using Topological Data Analysis (TDA) in a learning context, but robust topological information commonly comes in the form of a set of persistence diagrams, objects that by nature are uneasy to affix to a generic machine learning framework.

We introduce a vectorisation method for diagrams that allows to collect information from topological descriptors into a format fit for machine learning tools. Based on a few observations, the method is learned and tailored to discriminate the various important plane regions a diagram is set into. With this tool one can automatically augment any sort of machine learning problem with access to a TDA method, enhance performances, construct features reflecting underlying changes in topological behaviour. The proposed methodology comes with only high level tuning parameters such as the encoding budget for topological features. We provide an open-access, ready-to-use implementation and notebook.

We showcase the strengths and versatility of our approach on a number of applications. From emulous and modern graph collections to a highly topological synthetic dynamical orbits data, we prove that the method matches or beats the state-of-the-art in encoding persistence diagrams to solve hard problems. We then apply our method in the context of an industrial, difficult time-series regression problem and show the approach to be relevant.

Introduction

Topological Data Analysis (TDA) is a field dedicated to the capture and description of relevant geometrical or topological information from data. The use of TDA with standard machine learning tools has proved particularly advantageous in dealing with all sorts of complex data, meaning objects that are not or partly Euclidean, for instance graphs, time series, etc. The applications are abundant, from social network analysis, bio and chemoinformatics, to medical imaging and computer vision.

Through persistent homology, a multi-scale analysis of the geometric properties of the data, robust and stable information can be extracted. The resulting topological features are commonly computed in the form of a persistence diagram whose structure (an unordered set of points in the plane representing birth and death times for the features) does not easily fit the general machine learning input format. Therefore TDA is generally combined to machine learning by way of an embedding method for persistence diagrams.

Contributions. Our work is set in that trend: we introduce an automatic and learned vectorisation method designed to accurately reflect the inner variations of a given set of persistence diagrams. We showcase how, using this tool, integrating topological analysis into challenging learning problems leads to state-of-the-art results.

These contributions induce the following advantages: it allows for integration of TDA into a standard machine learning pipeline. There is little to no tuning, and a low level of knowledge of TDA is required, therefore the framework has a democratisation effect for learning on topological problems. It allows for univariate or multivariate topological representations of complex objects: graphs, time series, dynamical systems with interpretability benefits – connected to interpretability in TDA. Lastly it is a simple, efficient, swiss-knife embedding method that is of use in all sorts of learning problems and proving very efficient already.

Related work. A first line of combining persistence diagrams with machine learning is by designing a convenient vector representation. For instance it involves interpreting diagrams as images in (Adams et al. 2017), extracting topological signatures with respect to fixed points whose optimal position are learnt in (Hofer et al. 2017), a square-root transform of their approximated pdf in (Anirudh et al. 2016).

A second line introduces a specific kernel as the multi-scale kernel of (Reininghaus et al. 2015), the weighted Gaussian kernel of (Kusano, Hiraoka, and Fukumizu 2016) or the sliced Wasserstein kernel of (Carrière, Cuturi, and Oudot 2017). Those techniques have state-of-the-art behaviour on problems we will showcase, but for drawback they require another step for an explicit representation, and are known to scale poorly.

A recent orthogonal line of work has managed to directly combine the uneasy structure of persistence diagrams to neural networks architectures (Zaheer et al. 2017), (Carrière et al. 2019), with some interesting successes. Neural networks have immense benefits but are also heavy to deploy and hard to understand. They can always be paired with a representation method as in (Hofer et al. 2017).

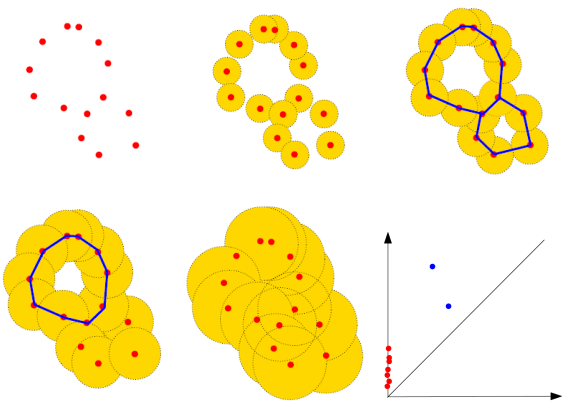

Figure 1: Example of a filtration by union of balls built on top of a 2-dimensional data set (red points) and its corresponding persistence diagram. As the radius of the balls increases, the connected components initially represented by each data point get merged; two cycles appears and disappears along the filtration. The connected components give rise to the red points on the vertical axis of the diagram as their birth time are all 0, and the cycles give rise to the two blue points.

Methodology

Persistent homology in TDA

Persistent homology plays a central role in TDA. It provides a rigorous mathematical framework and efficient algorithms to encode relevant multi-scale topological features of complex data, usually represented as point clouds or more complex geometric objects such as, e.g., time-series, graphs, 3D images,... - see (Edelsbrunner and Harer 2010; Boissonnat, Chazal, and Yvinec 2018) for a detailed introduction to persistent homology.

More precisely, persistent homology encodes the evolution of the topology of families of nested topological spaces $(F_\alpha)_{\alpha\in A}$, called filtrations, built on top of the data and indexed by a set of real numbers A that can be seen as scale parameters. For example, for a point cloud in Euclidean space, F_α can be the union of the balls of radius α centered on the data points - see Figure 1. Given a filtration $(F_\alpha)_{\alpha\in A}$ its topology (homology) changes as α increases: new connected components can appear, existing connected components can merge, cycles and cavities can appear or be filled, etc. Persistent homology tracks these changes, identifies features and associates, to each of them, an interval or lifetime from α_{birth} to α_{death}. For instance, a connected component is a feature that is born at the smallest α such that the component is present in F_α, and dies when it merges with an older connected component. The set of intervals representing the lifetime of the identified features is called the barcode of the filtration. As an interval can also be represented as a point in the plane with coordinates $(\alpha_{birth}, \alpha_{death})$, the set of points representing the intervals is called the persistence diagram of the filtration. The main advantage of persistence diagrams is that:

(i) they are proven to provide robust qualitative and quantitative topological information (Chazal et al. 2016) about the data;
(ii) since each point of the diagram represents a specific topological feature with its lifespan, they are easily interpretable as features;
(iii) from a practical perspective, persistence diagrams can be efficiently computed from a wide family of filtrations (The GUDHI Project 2015).
However, as persistence diagrams come as unordered set of points with non constant cardinality, they cannot be immediately processed as standard vector features in machine learning algorithms.

Automatic featurisation of persistence diagrams

Let $\mathcal{D}$ be the space of persistence diagrams. For ease of presentation we will not be considering points with infinite y-coordinates i.e. infinite lifetime, therefore in this work a persistence diagram D consists in a finite collection of points in $\mathbb{R}^2$.

We introduce a featurisation method for elements of $\mathcal{D}$, see Algorithm 1. Given a budget $b \in \mathbb{R}_+^*$ and diagram examples $D_1, \ldots, D_L$ for $L \in \mathbb{R}_+^*$, we use this fixed number of observations in order to tailor a vectorisation map with budget b for this space, that is we use $D_1, \ldots, D_L$ to produce a map $a : \mathcal{D} \to \mathbb{R}^b$. This map would ideally reflect the core topological variations of $\mathcal{D}$ in Euclidean space $\mathbb{R}^b$.

Algorithm 1: ATOL-featurisation

Data: Collection of persistence diagrams $(D_1, \ldots, D_L)$
parameters: budget $b \in \mathbb{N}^*$
Result: vectorisation map $a : \mathcal{D} \to \mathbb{R}^b$

1 Concatenate diagram collection into collection of points in $\mathcal{P} \subset \mathbb{R}^2$;
2 Run extraction algorithm on $\mathcal{P}$ (e.g. clustering algorithm) to produce centers $c_1, \ldots, c_b$ in $\mathbb{R}^2$ and deviations $e_1, \ldots, e_b$ in $\mathbb{R}_+^*$. Then define Laplacian contrast functions:
$l_k : \mathbb{R}^2 \to \mathbb{R}, d \mapsto \exp\left[-\frac{|c_k - d|_2}{e_k}\right]$;
3 Compute vectorisation map:
$a : \mathcal{D} \to \mathbb{R}^b, D \mapsto \left[\sum_{d\in D} l_k(d)\right]_{k\in[b]}$

To that effect we will use localisation variations between various elements of $D_1, \ldots, D_L$, that is we set to find regions of the plane where points in $D_1, \ldots, D_L$ seem to aggregate or disperse, and build a dedicated way to describe those regions. One generic way of doing it is using a clustering algorithm on the concatenated collection of points $\mathcal{P} := \bigcup_{i\in[L]} D_i$, and extract representative points per cluster.

In practice we run Lloyd's k-means algorithm (Lloyd 1982), extract clusters, centers and use them to produce what we call Laplacian contrast functions. For a given center $c \in \mathbb{R}^2$ associated with cluster $G \subset \mathcal{P}$, we compute cluster

deviation e and Laplacian contrast function $l : \mathbb{R}^2 \to \mathbb{R}$:

$$e := \sqrt{\sum_{d \in G} |d - c|_2^2} \tag{1}$$

$$l : d \mapsto \exp\big[- \frac{|c - d|_2}{e} \big]. \tag{2}$$

Therefore each center is associated a specific-range contrast function. The constrasts functions are then concatenated to form a vector reflecting topological information gathered from persistence diagrams.

Note that embedding map a is derived without knowledge of a learning task, its derivation is fully unsupervised. The representation is learned since it is data-dependent, but it is also agnostic to the task and only depends on getting a glimpse at a few persistence diagrams. This helps to localise important aggregation points (centers) and determine an average range of observation (deviations) from which to collect information (contrasts) on all diagrams.

This featurisation is close to a non-neural-network version of (Hofer et al. 2017) or to the codebook method of (Zielinski et al. 2018), but distinctly differs from these methods in the contrast maps introduced above, and those are key to our method's state-of-the-art results, see the Competitive TDA learning Section.

Automatic topological learning

In the context of a standard learning problem $\Omega := (X, y)$, the previous algorithm can be used to solve, help solve or provide some simplified topological understanding on elements $X \in \mathcal{X}$ of the problem.

This is the framework of Algorithm 2: after observing some elements $X_1, \ldots, X_L$ in $\mathcal{X}$ (in the context of a learning problem there always will be such a $L > 0$), after using persistent homology to derive collections of diagrams associated to those elements $D_1, \ldots, D_L$, Algorithm 1 provides an efficient way to produce a vectorised representation for space $\mathcal{X}$. Those features can then be used on their own or concatenated with other relevant features in learning schemes.

Algorithm 2: ATOL: Automatic Topologically-Oriented Learning

Data: Complex learning problem $\Omega := (X, y)$ with $X \in \mathcal{X}$ complex elements and y labels (available, partially available or hidden)
parameters: method $\Phi : \mathcal{X} \to \mathcal{D}$ yielding topological descriptors, $b, L \in (\mathbb{N}^*)^2$
Result: Enhanced learning problem $\widetilde{\Omega} := ((X, v(X)), y)$ with topological, euclidean features $v(X) \in \mathbb{R}^b$

1 Using a small number of observations $\Phi(X_1), \ldots, \Phi(X_L)$, produce a vectorisation map a with Algorithm 1;
2 For all elements of the learning problem, compute their topological descriptors $\Phi(X)$ and featurisation through map a, so that each element X is described by $v(X) = a(\Phi(X)) \in \mathbb{R}^b$.

Competitive TDA learning

In this section we demonstrate the advantages of our approach. By addressing assorted and challenging applications (a collection of graphs, dynamical orbits and time-series learning problems), we show the ATOL framework to be not only competitive and state-of-the-art, but also strongly efficient with respect to compacity and tuning, easy to use with high automaticity while also allowing for interpretability with respect to the underlying topological features. Those characteristics make it a sort of swiss-knife method, easily deployed and operated for any sort of problem where one would project a topological analysis foray.

Graph classification problems

Learning problems involving graph data are receiving an extraordinary amount of interest, and we now showcase the greatest value of our method: it is highly competitive.

Recently (Carrière et al. 2019) have introduced a powerful way of extracting topological information from graph structures. They make use of heat kernel signatures (HKS) for graphs (Hu, Rustamov, and Guibas 2014), a spectral family of signatures (with diffusion parameter $t > 0$) whose topological structure can be encoded in the extended persistence framework, yielding four types of topological features with exclusively finite persistence. On both of those points we refer to Sections 4.2 and 2 from (Carrière et al. 2019). Therefore for each graph and HKS diffusion time t the resulting topological descriptor from (Carrière et al. 2019) are four persistence diagrams with all finite coordinates.

We leverage those topological descriptors and for fair comparison use the same benchmarks to demonstrate the efficiency of our method. It includes small and large sets of graphs (`MUTAG` has 188 graphs, `REDDIT12K` has 12000), small and large graphs (`IMDB-M` has 13 nodes on average, `REDDIT5K` has more than 500), dense and sparse graphs (`FRANKENSTEIN` has around 12 edges per nodes, `COLLAB` has more than 2000), binary and multi-class problems (`REDDIT12K` has 11 classes), and graphs of different nature (social networks, chemoinformatics, bioinformatics). All datasets can be found in the repository (Kersting et al. 2016).

For the entire set of probems we use the same two HKS diffusion times to be .1 and 10, fueling the extended graph persistence framework and resulting in 8 persistence diagrams per graph. For budget we choose $b = 80$ for all experiments. We discard and make no use of graph attributes on edges or vertices that some dataset exhibit, and no other sort of features are collected, i.e. Algorithm 2 here simply consists in reducing the original problem from Ω to $\widetilde{\Omega} := (v(X), y)$ with $v(X) \in \mathbb{R}^{80}$. To compute the embedding map a (Algorithm 1) we use all available diagrams in the training set (without supervision). The resulting embedding is empirically stable as soon as the dataset contains more than a few dozen elements.

To give an approximate value to our method's worth

in this learning context, we evaluate the featurisation for classification purposes using the standard `scikit-learn` (Pedregosa et al. 2011) random-forest classification tool with 100 trees and all other parameters set as default. On each problem we perform a 10-fold evaluation and average the resulting accuracies. Mean accuracies and standard deviations over 10 such experiments are shown Table 1.

methods problems	RetGK1	RetGK11	FGSD	GCNN	PersLay	ATOL
REDDIT5K	56.1(0.5)	55.3(0.3)	47.8	52.9	56.6(0.3)	**58.5**(0.2)
REDDIT12K	**48.7**(0.2)	47.1(0.3)	—	46.6	47.7(0.2)	44.5(0.1)
COLLAB	81.0(0.3)	80.6(0.3)	80.0	79.6	76.4(0.4)	**83.9** (0.1)
IMDB-B	71.9(1.0)	72.3(0.6)	73.6	73.1	70.9(0.7)	**74.4** (0.7)
IMDB-M	47.7(0.3)	48.7(0.6)	**52.4**	50.3	48.7(0.6)	47.9 (0.6)
BZR	—	—	—	—	**87.2**(0.7)	74.4 (1.9)
COX2	80.1(0.9)	81.4(0.6)	—	—	**81.6**(1.0)	58.6 (1.4)
DHFR	81.5(0.9)	**82.5**(0.8)	—	—	81.8(0.8)	80.3 (0.7)
MUTAG	90.3(1.1)	90.1(1.0)	**92.1**	86.7	89.8(0.9)	86.9 (1.3)
PROTEINS	75.8(0.6)	75.2(0.3)	73.4	**76.3**	74.8(0.3)	70.0 (0.4)
NCI1	**84.5**(0.2)	83.5(0.2)	79.8	78.4	72.8(0.3)	78.9 (0.3)
NCI109	—	—	**78.8**	—	71.7(0.3)	77.5 (0.5)
FRNKNSTN	—	—	—	—	70.7(0.4)	**72.8** (0.4)

Table 1: Mean accuracy and standard deviation over ten 10-folds for RetGK1 and RetGK11 (Zhang et al. 2018), FGSD (Verma and Zhang 2017) (single 10-fold), GCNN (Xinyi and Chen 2019) (single 10-fold), PersLay (Carrière et al. 2019) and our method. Bold fonts indicate the best scores.

Those results are state-of-the-art, gaining two to three points on the next best contender on two of the more difficult datasets (`REDDIT5K` and `COLLAB`). Overall all results represent solid performances except for one surprising miss on small problem `COX2`.

The competitors shown here are tailored to graph problems (Perslay excepted), with two graph kernel methods (RetGK1, RetGK11), one graph embedding method (FGSD) and one capsule neural network method (GCNN). (Hofer et al. 2017) also report on `REDDIT5K` and `REDDIT12K`, with mean accuracies of 55.5 and 44.5 for 10 cross-validation runs.

Computations are run on a laptop (i5-7440HQ 2.80 GHz CPU), and Algorithm 1 takes about 1 seconds on small `MUTAG` dataset, and 8 seconds on average size `IMDB-B` dataset (single fold). The results presented here are easily accessible (requiring open source C++/Python library Gudhi (The GUDHI Project 2015), Python library (Pedregosa et al. 2011)) and reproducible with the public repository github.com/martinroyer/atol.

We now stress that all those experiments were run with the exact same configuration for Algorithm 2. The budget was selected as a round number: 10 centers for each of the four diagram types of extented persistence, for each filtration, hence $b = 80$. Moreover, this parameter is not pivotal to the performances reported, as they are stable across a wide range of such parameter. Figure 2 shows the variation in performances for four problems as the budget is increased. It is apparent that the budget selection, Algorithm 2's main parameter, is not crucial, and that the procedure is performant even for very compact representations with only a couple centers par diagram type.

Overall, the simplicity and absence of tuning hint at robustness and good generalisation power.

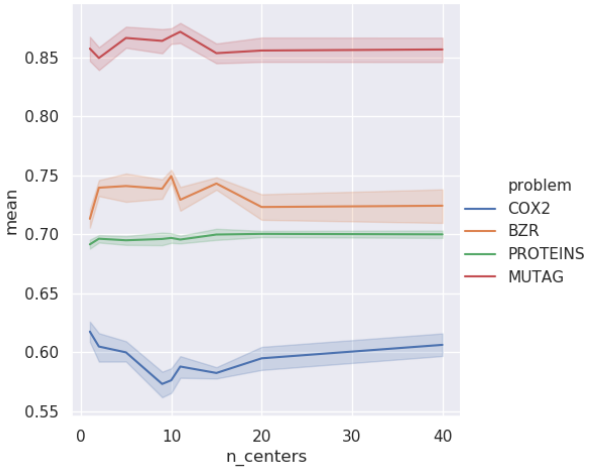

Figure 2: Accuracies and standard deviations over ten 10-fold for four of the graph problems, as the number of centers per diagram type per filtration is increased. The resulting budget is $b = 8 \times$ `n_centers` as there are 4 diagrams type and 2 filtrations. The chosen value for all graph experiments is `n_centers` $= 10$.

Reflecting topological variations in dynamical orbits

(Adams et al. 2017) use a synthetic, discrete dynamical system (used to model flows in DNA microarrays) with the following property: the resulting chaotic trajectories exhibit distinct topological characteristics depending on a parameter $r > 0$. The dynamical system is:

$$\begin{aligned} x_{n+1} &:= x_n + r y_n(1 - y_n) && \mod 1 \\ y_{n+1} &:= y_n + r x_{n+1}(1 - x_{n+1}) && \mod 1 \end{aligned}$$

With random initialisation and five different parameters $r \in \{2.5, 3.5, 4, 4.1, 4.3\}$, a thousand iterations per trajectory and a thousand orbits per parameter, a datasets of five thousand orbits is constituted and commonly used for evaluating topological methods. The problem of classifying this datasets in accordance to their underlying parameter is rather uneasy and challenging.

We apply our framework on persistence diagrams from the AlphaComplex filtrations in dimensions 0 and 1 with a total budget of $b = 80$. After a learning phase with a 70/30 split, we measure accuracy over a hundred runs. This yields a 84.2 (0.8) mean accuracy and deviation. In the exact same experimental setup, the best competitive methods have obtained the following mean accuracies: 72.38 (2.4) (Reininghaus et al. 2015), 76.63 (0.7) (Kusano, Hiraoka, and Fukumizu 2016), 83.6 (0.9) (Carrière, Cuturi, and Oudot 2017), 85.9 (0.8) (Le and Yamada 2018), 87.7 (1.0) (Carrière et al. 2019). Therefore our results are also competitive for this problem.

Let us now restrict the problem to a two-class problem, i.e. only select orbits generated with parameters $r \in$

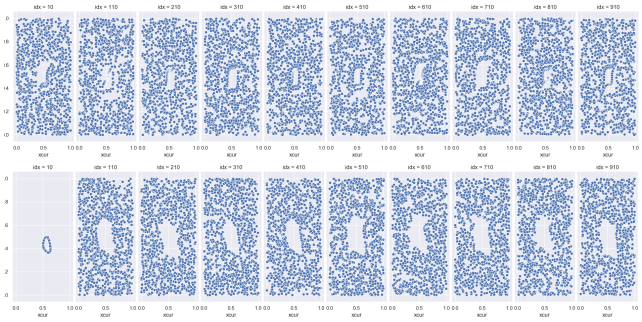

Figure 3: Example of synthetised orbits (x and y coordinates in the flat torus $[0, 1]^2$) with parameter 4.0 (top row) and 4.1 (bottom row).

$\{4.0, 4.1\}$. Figure 3 shows a few such orbits. For orbits generated with parameter $r = 4.1$, it happens that the initialisation is close to an attractor point, that gives it the special shape as in the leftmost orbit.

The persistence diagrams generated for each orbit are featurised through Algorithm 1 and Figure 4 shows such feature for a given center. The spikes match the specific orbits with special shape mentioned above, but the overall mean change in value reflects the underlying change in topological structure. Each such features allows for univariate representation of the original data, and the combine use of ten such features makes for a 88% accuracy score in this context. As centers are locally fixed within the $\mathbb{R}^2$ plane, one can infer which elements within the sets are drawn towards which centers and derive topological information with respect to the original object without any supervision. Lastly, after learning, center importance in learning task may also be determined (Figure 4) and provide additional understanding.

Topological score for time series, an industrial application

Finally we present an industrial application for time series, in a case where the learning problem is hard and no obvious solutions are to be found.

This dataset consists in the following experiments: using commercially available simulator of a Japenese city road circuit course, about a hundred subjects are monitored (RRI data sampled at 4Hz) for a 80 minutes drive that includes two periods of high-speed driving at the beginning and at the end of the experiment, and a low-speed driving period in the middle designed to induce sleepiness. For each experiment, an expert annotation (labeled NEDO score) produced from observing the driver is made available, indicating sleepiness on a 1 to 5 class scale. We show four such experiments in Figure 5 (the RR-intervals have been normalised).

This problem of retrieving the sleepiness level based on RRI levels is hard and ill-posed: there are strong individual differences in perceived reaction to a given situation, a single experiment per subject to learn behaviour from, and apparent noise or absence of signal in annotations, see e.g. subject 3 in Figure 5. Nevertheless we propose to use

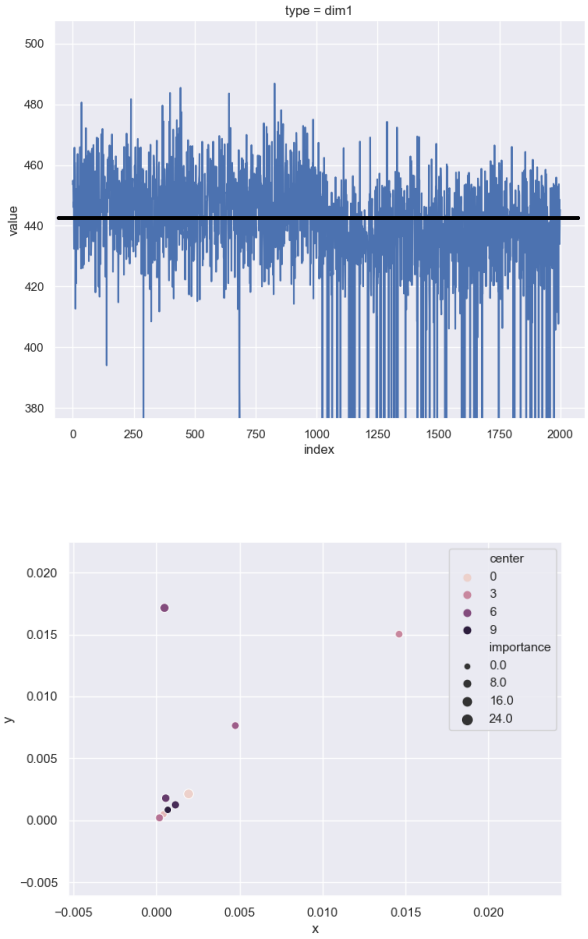

Figure 4: Top: for the 2-class problem, feature value for each orbit with respect to a given center. Indices 0 to 999 are for orbits with parameter 4.0, indices 1000 to 1999 for orbits with parameter 4.3. The black line is immaterial, and drawn for visualisation purposes. Bottom: for the 5-class problem, example centers for homological dimension 1 drawn in the plane and their respective importance in the initial classification task.

the ATOL framework to produce features meant reflect the sleepiness level in subjects based on RRI variations. The intent is that even though this will poorly reflect the latent sleepiness level, this could be enough to allow to catch jumps in the perceived attention level.

The framework can readily be applied to time series in any given dimension and used to produce topological features. For this application we will follow a classical path: *(i)* use a sliding window decomposition on the time-series, *(ii)* use a time-delay embedding to transform said window into a point cloud, *(iii)* apply persistent homology analysis (we will use DTM-filtration) to produce persistence diagrams and *(iv)* vectorise persistence diagrams using Algorithm 1. We concatenate those features with the mean and standard deviation statistics on the sliding-window. As for learning, we compute a learner based on other individuals' features regressed to their NEDO scores, and use it to generate a score based on ATOL features (see middle and bottom row in Figure 6).

Although this score imperfectly reflects the underlying NEDO score for a given patient, is can still have some uses. We set to detect two jumps on this topologically-augmented score using a Gaussian Kernel. We also compute a regressor based on the standard features without additional topological features, for comparison purposes, and also detect two jumps on this standard score.

Figure 6 shows two example results of our analysis. Each panel (top and bottom) consists in three time-series: the (hidden) NEDO score (top row), the ATOL-score computed from a regressor based on topological features (middle row), and a standard score computed from a regressor based solely on standard features (bottom row). The changes of colour from blue to red and to blue indicates the changes in the experimental design for the driving simulation, i.e. the red portion indicates low-speed driving whereas the blue portions indicate high-speed driving periods. The black dotted lines indicate jumps detected from the ATOL representation, whereas the red dotted lines indicate jumps detected from the standard representation. In the top panel, the two series of jumps are concomitant, and almost an exact match to the underlying changes in the experimental design. In the bottom panel, an improvement over the standard score is caught with the ATOL score that better reflects the changes in latent NEDO score for this subject, two the point that the detected jumps are an exact match for the changes in experimental conditions. Overall, the ATOL score has less spikes and more regularity than the standard score, which is expected as the topological features are extracted posterior to a time-delay embedding procedure.

Figure 5: Measured drowsiness and annotated NEDO score for four subjects.

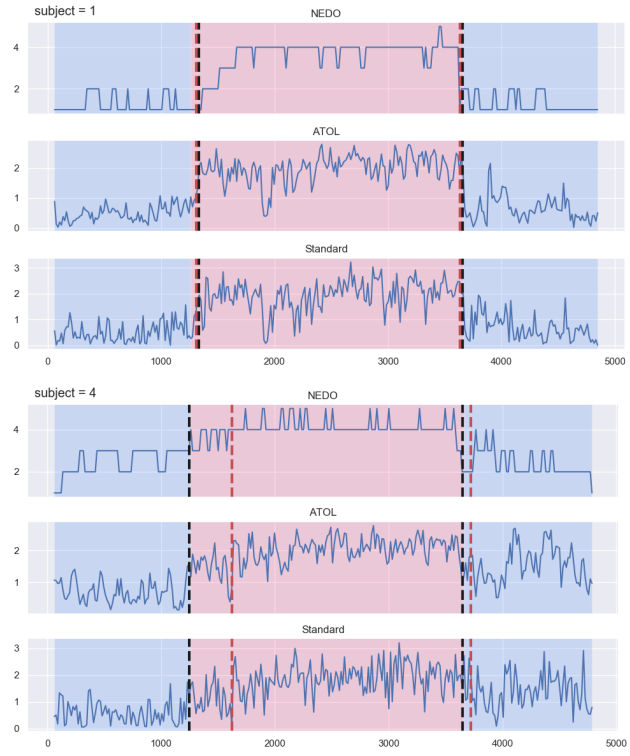

Figure 6: Results of NEDO score (top row) regression and of a 2-jumps detection procedure, from standard features alone (bottom row) and ATOL features (middle row). Red zones indicates low-speed section of the simulation, blue zones indicates high-speed section.

References

Adams, H.; Emerson, T.; Kirby, M.; Neville, R.; Peterson, C.; Shipman, P.; Chepushtanova, S.; Hanson, E.; Motta, F.; and Ziegelmeier, L. 2017. Persistence images: a stable vector representation of persistent homology. *Journal of Machine Learning Research* 18(8).

Anirudh, R.; Venkataraman, V.; Ramamurthy, K. N.; and Turaga, P. 2016. A riemannian framework for statistical analysis of topological persistence diagrams. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1023–1031.

Boissonnat, J.-D.; Chazal, F.; and Yvinec, M. 2018. *Geometric and Topological Inference*, volume 57. Cambridge University Press.

Carrière, M.; Chazal, F.; Ike, Y.; Lacombe, T.; Royer, M.; and Umeda, Y. 2019. PersLay: A Simple and Versatile Neural Network Layer for Persistence Diagrams. *arXiv e-prints* arXiv:1904.09378.

Carrière, M.; Cuturi, M.; and Oudot, S. 2017. Sliced Wasserstein kernel for persistence diagrams. In *International Conference on Machine Learning*, volume 70, 664–673.

Chazal, F.; de Silva, V.; Glisse, M.; and Oudot, S. 2016. *The structure and stability of persistence modules*. SpringerBriefs in Mathematics. Springer.

Edelsbrunner, H., and Harer, J. 2010. *Computational Topology: An Introduction*. AMS.

Hofer, C.; Kwitt, R.; Niethammer, M.; and Uhl, A. 2017. Deep learning with topological signatures. In *Advances in Neural Information Processing Systems*, 1634–1644.

Hu, N.; Rustamov, R.; and Guibas, L. 2014. Stable and informative spectral signatures for graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2305–2312.

Kersting, K.; Kriege, N. M.; Morris, C.; Mutzel, P.; and Neumann, M. 2016. Benchmark data sets for graph kernels. http://graphkernels.cs.tu-dortmund.de.

Kusano, G.; Hiraoka, Y.; and Fukumizu, K. 2016. Persistence weighted Gaussian kernel for topological data analysis. In *International Conference on Machine Learning*, volume 48, 2004–2013.

Le, T., and Yamada, M. 2018. Persistence Fisher kernel: a Riemannian manifold kernel for persistence diagrams. In *Advances in Neural Information Processing Systems*, 10027–10038.

Lloyd, S. 1982. Least squares quantization in pcm. *IEEE Trans. Inf. Theor.* 28(2):129–137.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Reininghaus, J.; Huber, S.; Bauer, U.; and Kwitt, R. 2015. A stable multi-scale kernel for topological machine learning. In *IEEE Conference on Computer Vision and Pattern Recognition*.

The GUDHI Project. 2015. *GUDHI User and Reference Manual*. GUDHI Editorial Board.

Verma, S., and Zhang, Z.-L. 2017. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, 88–98.

Xinyi, Z., and Chen, L. 2019. Capsule graph neural network. In *International Conference on Learning Representations*.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R.; and Smola, A. 2017. Deep sets. In *Advances in Neural Information Processing Systems*, 3391–3401.

Zhang, Z.; Wang, M.; Xiang, Y.; Huang, Y.; and Nehorai, A. 2018. RetGK: Graph Kernels based on Return Probabilities of Random Walks. In *Advances in Neural Information Processing Systems*, 3968–3978.

Zielinski, B.; Lipinski, M.; Juda, M.; Zeppelzauer, M.; and Dlotko, P. 2018. Persistence Codebooks for Topological Data Analysis. *arXiv e-prints* arXiv:1802.04852.