University of Tennessee, 18 November 2021

Giotto – An open-source ecosystem for Topological Machine Learning

Umberto Lupo, EPFL

My awesome collaborators


Kathryn Hess


Guillaume Tauzin


Anibal Medina-Mardones


Alberto Dassatti


Julian Burella Pérez


Wojciech Reise


Lewis Tunstall


Matteo Caorsi

EPFL

ReDS
Reconfigurable & embedded Digital Systems

L2F

INAIT

Sponsored by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Innosuisse – Swiss Innovation Agency

… and many **community** contributors!

Plan

1. Intro: **Topology in data analysis**
2. Range of **applicability**
3. Integration with **machine learning** and the  projects
4. Hands-on session & audience questions

What is Topological Data Analysis?

- **Topological Data Analysis/Machine Learning (TDA/TML)** apply modern **computational algebraic topology** to data problems
- Main points of focus:
  1. Extract **succinct topological features ("the shape of data")** from discrete datasets by leveraging their **geometry**
  2. Multi-scale approach – describe the emergence and disappearance of topological features at **all scales at once**.
  3. Needs very little – often only **distance-like matrices** or **graphs** with edge and/or node weights, or greyscale (2-D or *N*-D) **images**. Leading to **wide applicability**.
- Also worth mentioning:
  - Suitable for very **high-dimensional** datasets.
  - **Robustness** to perturbations – ensured by stability theorems.
  - Impressive **scalability** improvements in recent years.

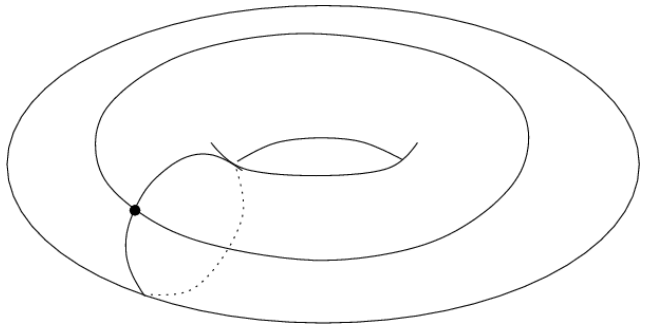
- **Topology** studies the **properties** of a **geometric object** that are **preserved under continuous deformations**, such as stretching, twisting, crumpling and bending, but not tearing or gluing.
  - Here, the surfaces of a coffee mug and of a donut are equivalent…
  - … but neither are equivalent to a sphere!

- **Algebraic** topology (AT) handles these properties via linear algebra.

  Example properties:
  - Connectedness
  - Number of holes, voids, …
- **Beyond continuous shapes**: AT can also deal with **purely combinatorial** objects such as graphs and their higher-dimensional counterparts, or grid-like data.

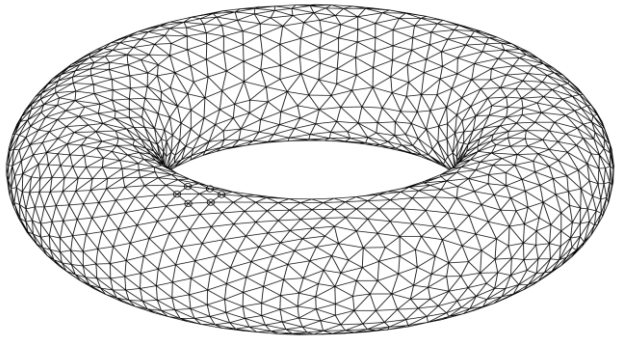
- **Computational AT** is AT's algorithmic division
- Example "old school" recipe starting from a **smooth object** (manifold):
  1. **Discretize** via a **mesh/triangulation**.
  2. **Compute shape signatures** of the discrete approximation (number of connected components, number of holes and voids, etc.).
  3. If the mesh is "sufficiently fine", its shape signatures are guaranteed to be the same as for the original continuous object.
  4. Use these shape signatures to **distinguish** between, **classify** objects, etc.

Hollow torus



Discretize

Triangulation



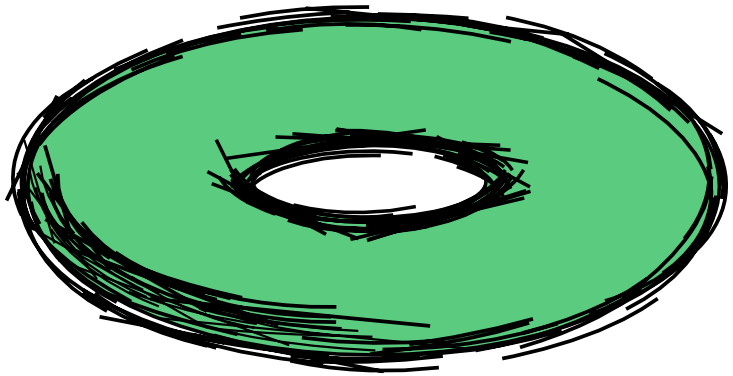
Compute

Shape signatures
"**Betti numbers**" $\boldsymbol{\beta_k}$

$\boldsymbol{\beta_0} = \mathbf{1}$ connected component
$\boldsymbol{\beta_1} = \mathbf{2}$ "independent" 1d holes
$\boldsymbol{\beta_2} = \mathbf{1}$ void

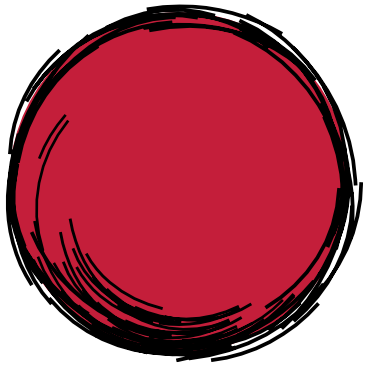
Topological features
"**Betti numbers**" $\boldsymbol{\beta_k}$

Torus



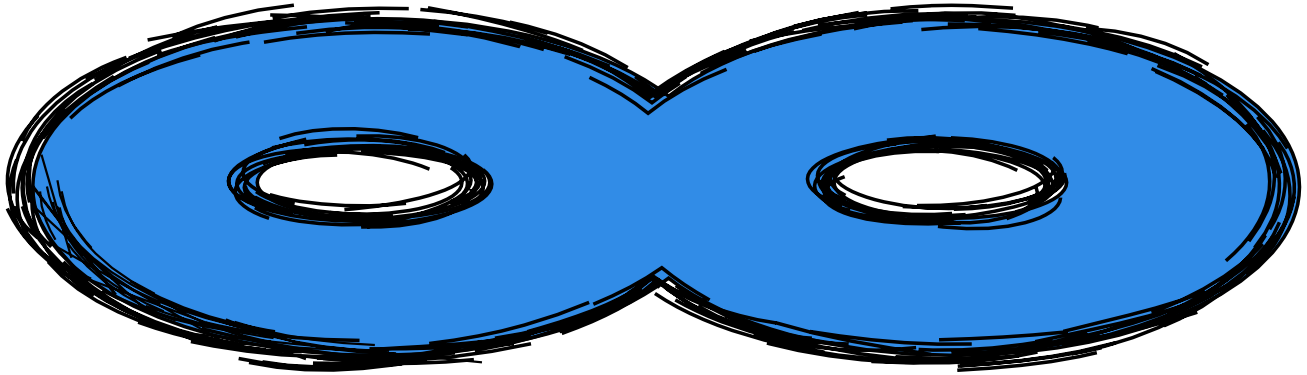
$\boldsymbol{\beta_0} = \mathbf{1}$ connected component
$\boldsymbol{\beta_1} = \mathbf{2}$ holes
$\boldsymbol{\beta_2} = \mathbf{1}$ void

Sphere



$\boldsymbol{\beta_0} = \mathbf{1}$ connected component
$\boldsymbol{\beta_1} = \mathbf{0}$ holes
$\boldsymbol{\beta_2} = \mathbf{1}$ void

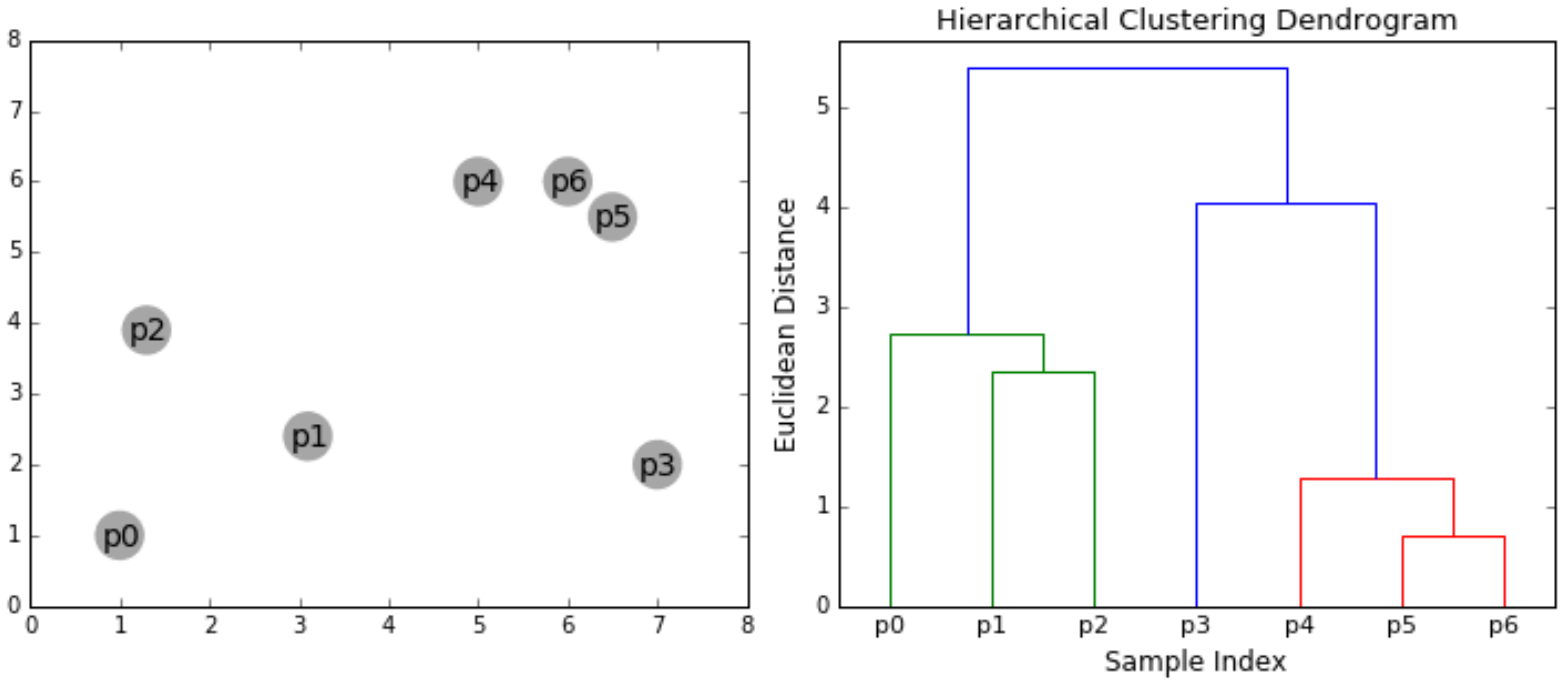
Two glued tori



$\boldsymbol{\beta_0} = \mathbf{1}$ connected component
$\boldsymbol{\beta_1} = \mathbf{4}$ holes
$\boldsymbol{\beta_2} = \mathbf{1}$ void

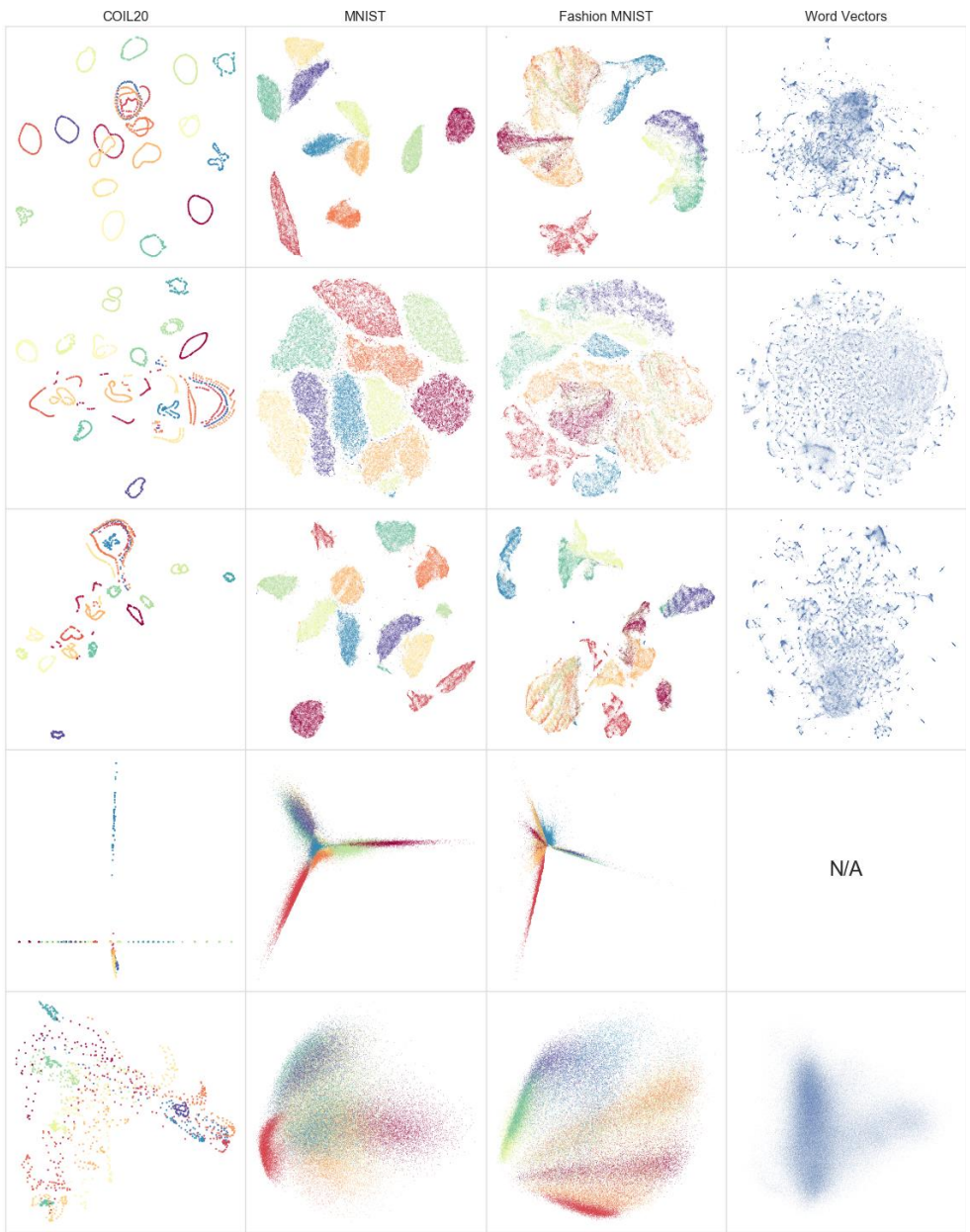
TDA in the mainstream? [1/2]

- Looking at 0-dimensional Betti numbers is looking at **connected components**
- So **(single-linkage) hierarchical clustering** is **“0-dimensional multi-scale TDA”**
- Vanilla single-linkage clustering is brittle. But persistence ideas + DBSCAN lead to HDBSCAN*, which is robust and fast!



Source: https://dashee87.github.io/data%20science/general/Clustering-with-Scikit-with-GIFs/

TDA in the mainstream? [2/2]



UMAP: a state-of-the-art algorithm for non-linear dimensionality reduction

L. McInnes, J. Healy and J. Melville. [UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction]

Published: 03 December 2018

Dimensionality reduction for visualizing single-cell data using UMAP

Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel W H Kwok, Lai Guan Ng, Florent Ginhoux & Evan W Newell

Nature Biotechnology **37**, 38–44(2019) | Cite this article

49k Accesses | **500** Citations | **282** Altmetric | Metrics

Article | Open Access | Published: 24 March 2020

Dimensionality reduction by UMAP to visualize physical and genetic interactions

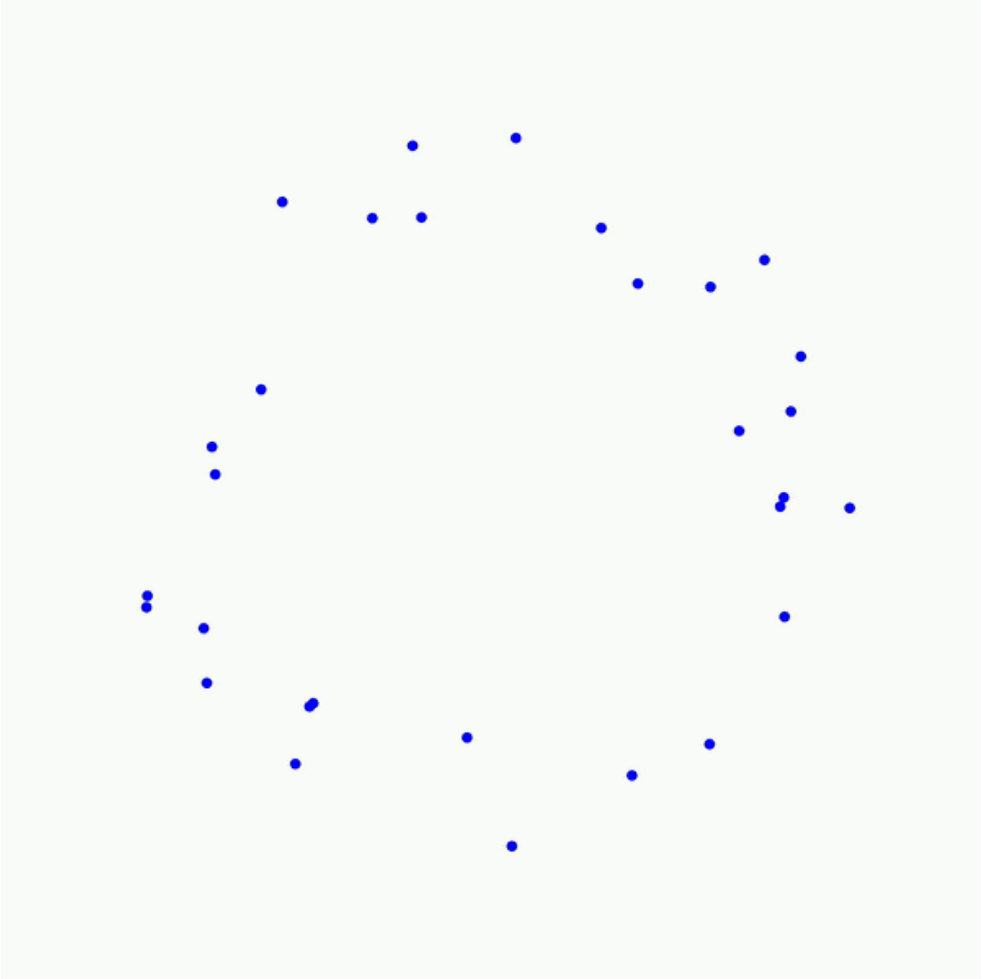
Michael W. Dorrity, Lauren M. Saunders, Christine Queitsch, Stanley Fields & Cole Trapnell

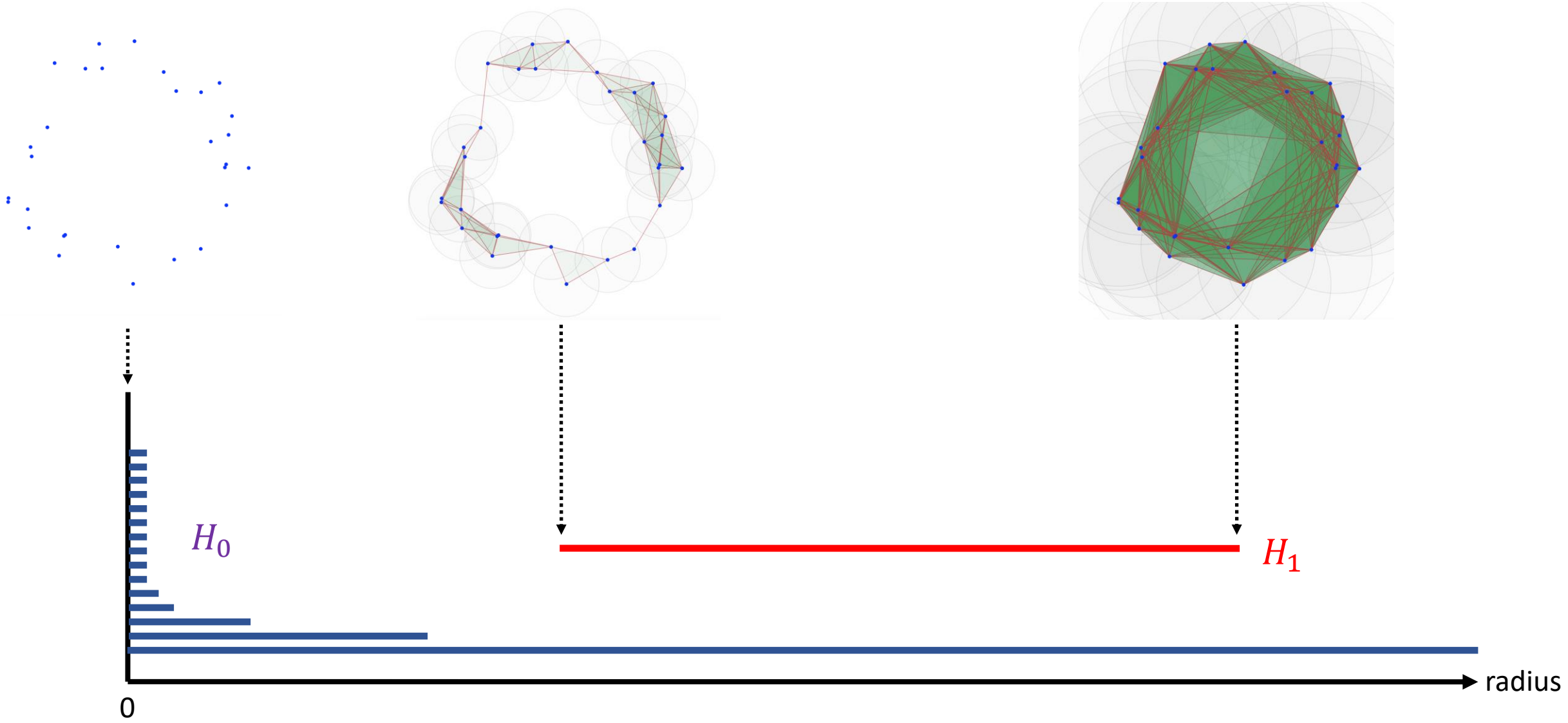
Nature Communications **11**, Article number: 1537 (2020) | Cite this article

7459 Accesses | **7** Citations | **65** Altmetric | Metrics

Persistent Homology

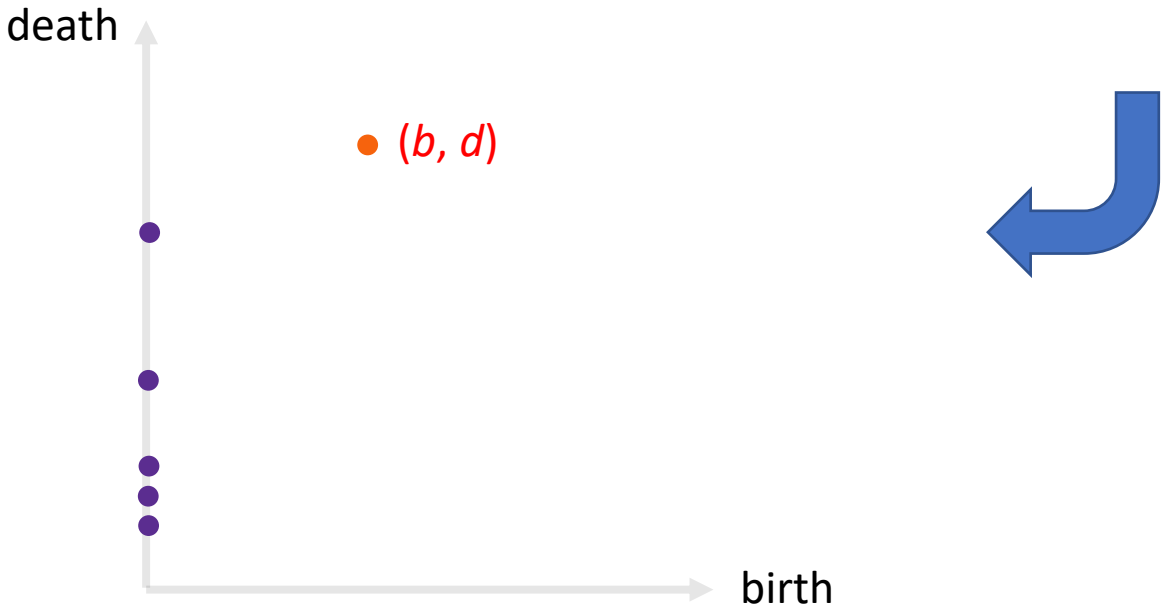
- **Simplicial complex**: generalization of a graph containing triangles, tetrahedra, ... (called **simplices**)
- TDA recipe:
  - make **one simplicial complex per parameter value r** according to a rule
  - Keep track of birth and death of topological features (connected comps, holes, voids, …)
- *Main Example of rule*: r is radius of ball grown around each point, edges appear when balls intersect, simplices are cliques ("**Vietoris-Rips**")
- Easy to go beyond Euclidean distance! **Dissimilarity** between vertices in a network, genetic distance for sequence data, …





"**Barcode**": bars start when topological features appear and end when they disappear.

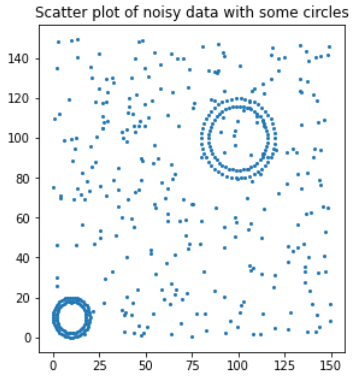
From barcode to **persistence diagram**: plot **birth-death values** as **2D coordinates**

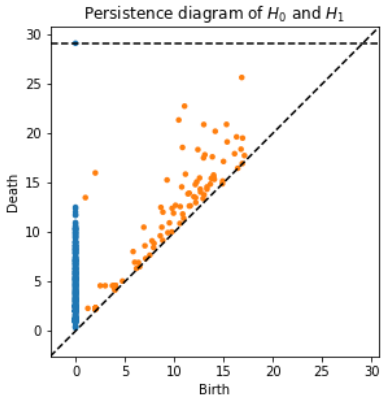


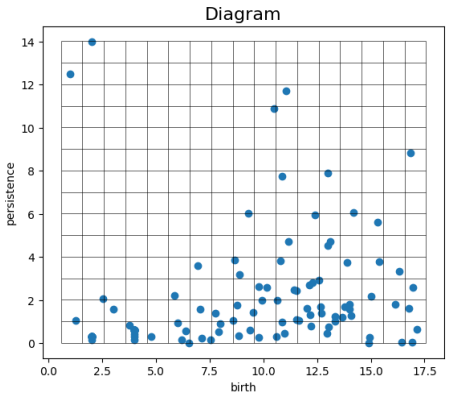
- Persistence diagrams are incredibly **succinct** data summaries
- **Distances** between diagrams exist with good statistical properties (**stability**). **Bottleneck**, **Wasserstein** (optimal transport)
- Diagrams are (**multi-**)**sets**, and interesting set functions are difficult to describe via conventional array-based computation. (An exception: **Shannon entropy**.)
- One solution: **vectorize** diagrams & feed resulting vectors to arbitrary functions
- **Inner product** structure can be used to define several **kernels and distances**

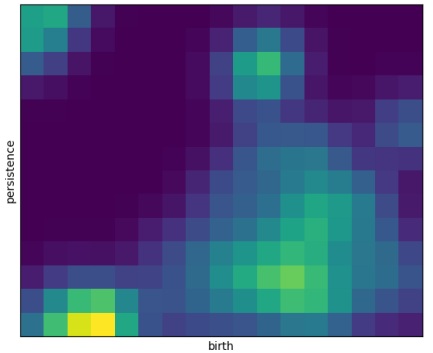
!

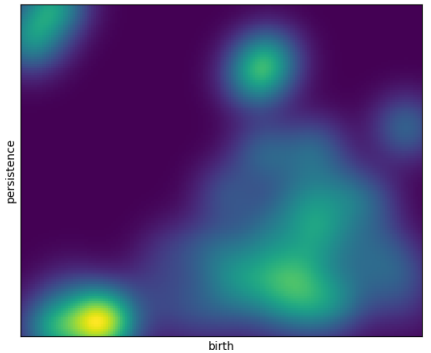
Example: **Persistence Images**







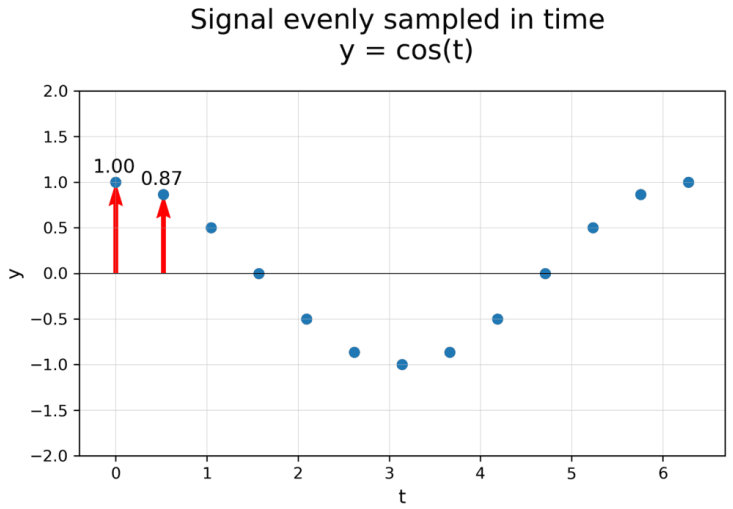


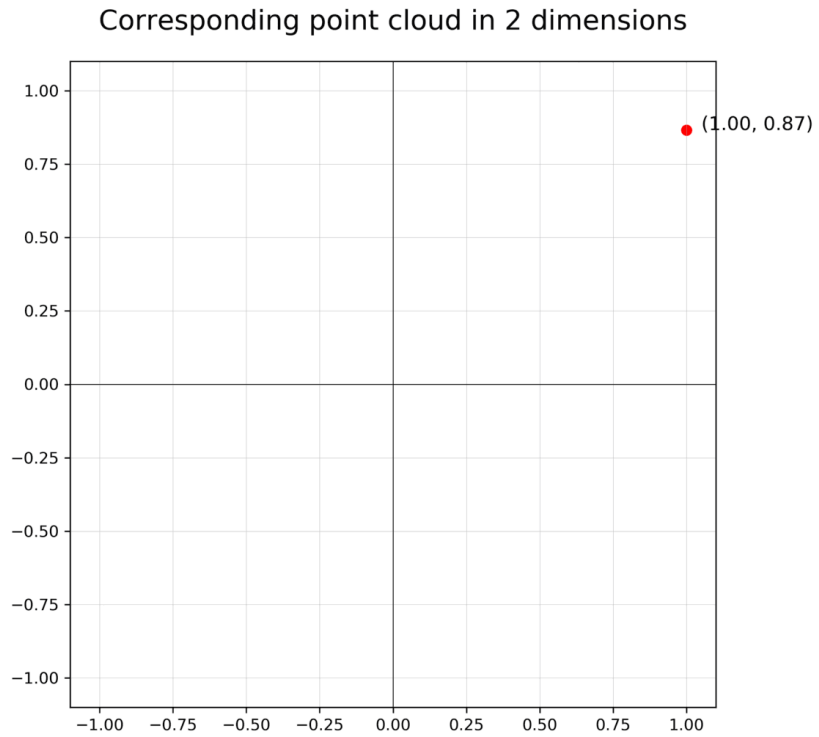


Source: https://persim.scikit-tda.org/en/latest/notebooks/Persistence%20images.html

Neat application: time series

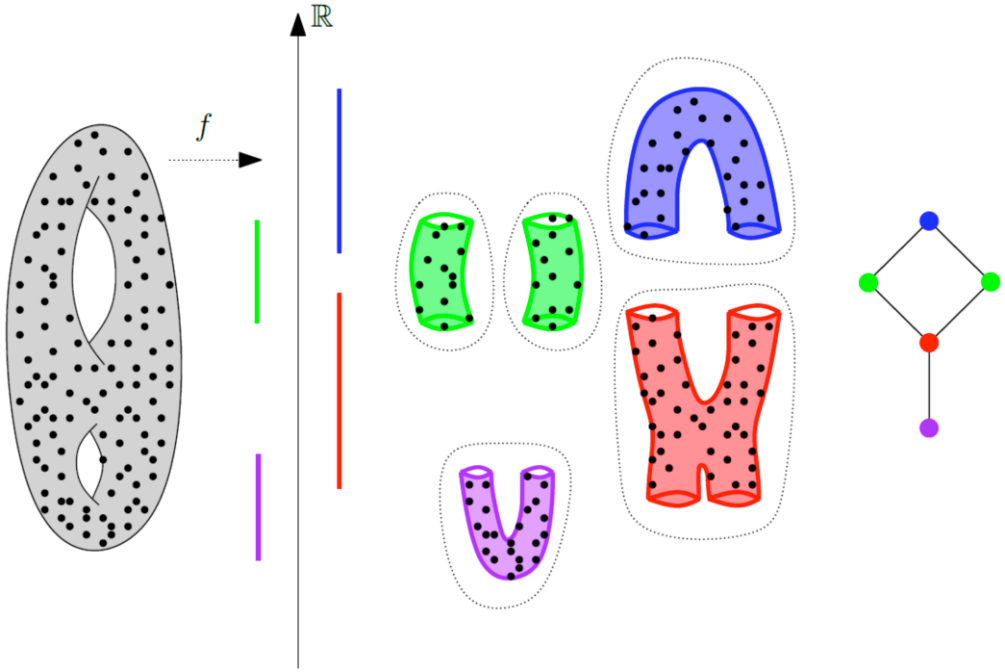
- "**Takens embedding**" technique: time series → point cloud → topological feature extraction
- **Periodicity** corresponds to **loops** in the embedding!



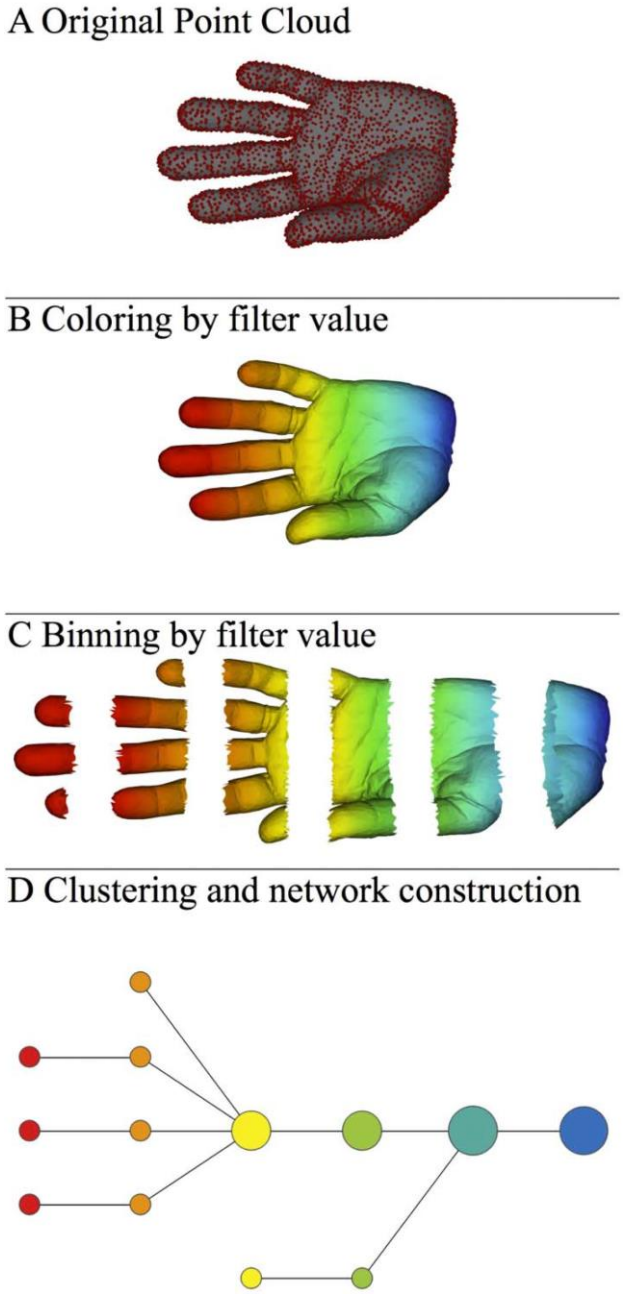


The Mapper algorithm [1/2]

- A popular TDA algorithm for **data exploration and discovery**
- Uses a particular “filter” on the data and partial clustering to obtain a skeletonized **topological summary** of the data
- Output is a **graph**:
  - each node is a cluster
  - there is an edge between two overlapping clusters



Here, filter function *f* is projection on one coordinate axis



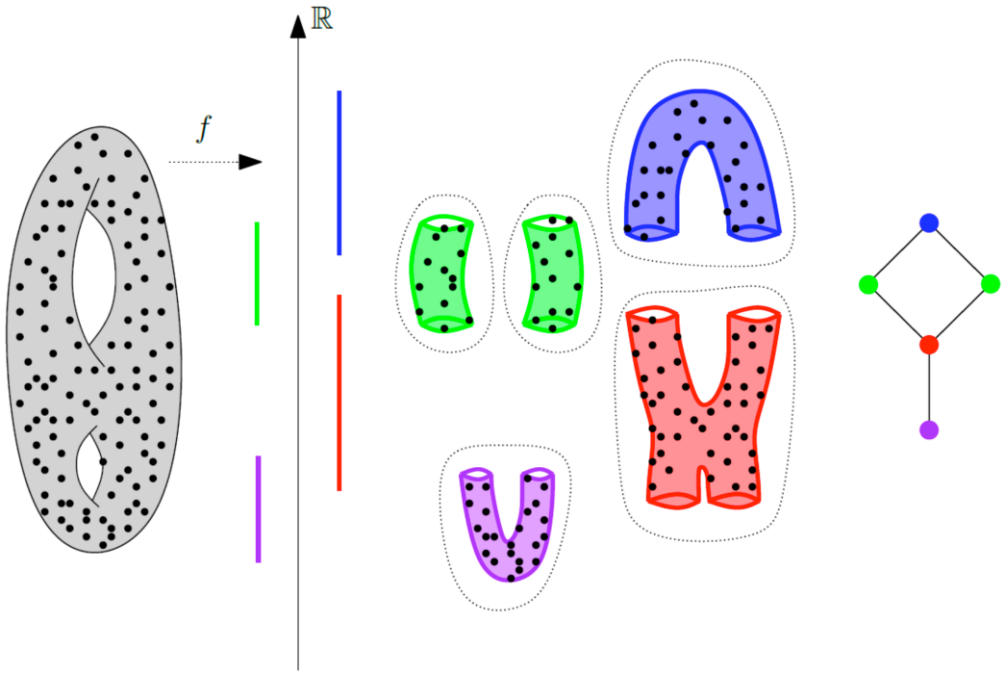
Extracting insights from the shape of complex data using topology

P. Y. Lum[1], G. Singh[1], A. Lehman[1], T. Ishkanov[1], M. Vejdemo-Johansson[2], M. Alagappan[1], J. Carlsson[3] & G. Carlsson[1,4]

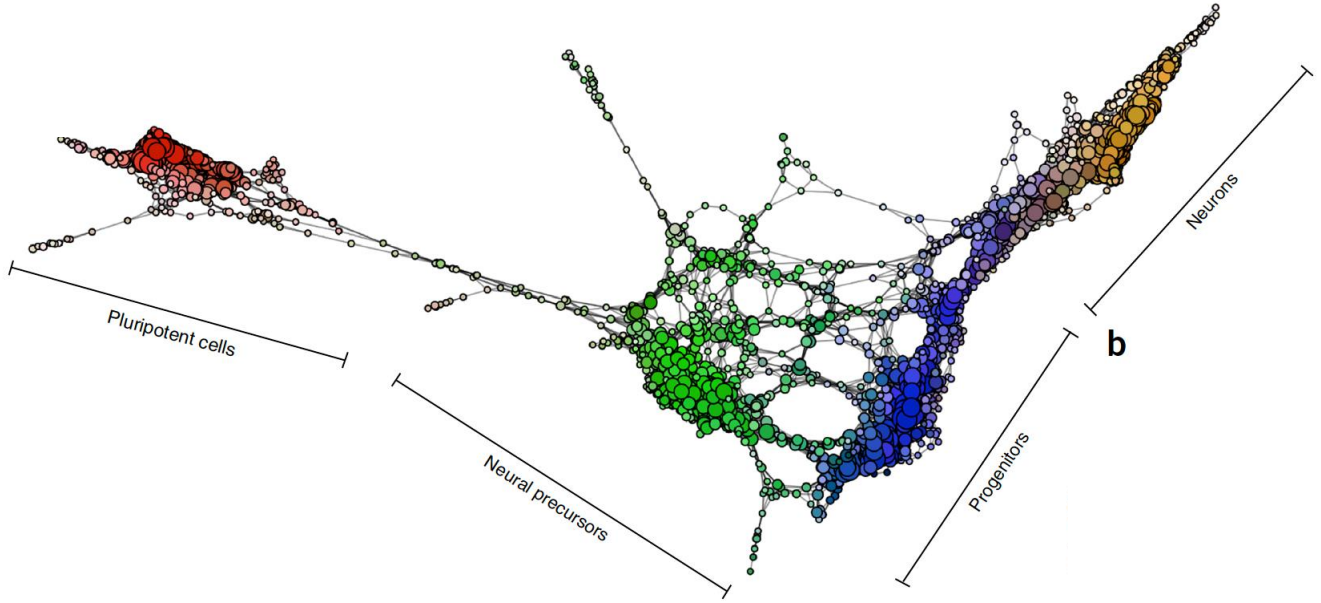
[1]Ayasdi Inc., Palo Alto, CA, [2]School of Computer Science, Jack Cole Building, North Haugh, St. Andrews KY16 9SX, Scotland, United Kingdom, [3]Industrial and Systems Engineering, University of Minnesota, 111 Church St. SE, Minneapolis, MN 55455, USA, [4]Department of Mathematics, Stanford University, Stanford, CA, 94305, USA.

The Mapper algorithm [2/2]

- Filter functions are **arbitrary** and can be **multi-dimensional**! Clustering algo also arbitrary.
- Can inject **domain knowledge** into these choices.
- Example: score of an **outlier detection algorithm** as a filter!
- Other filters worth trying: **dimensionality reduction** algos (PCA, t-SNE, UMAP, …)



Here, filter function f is projection on one coordinate axis



Single-cell topological RNA-seq analysis reveals insights into cellular differentiation and development

Abbas H Rizvi[1,2,6], Pablo G Camara[3,4,6], Elena K Kandror[1,2], Thomas J Roberts[1,2,4], Ira Schieren[2,5], Tom Maniatis[1,2] & Raul Rabadan[3,4]

Topology and the ML workflow

Our objective: Place topological learning algorithms firmly *alongside* established machine learning techniques

ML ethos: Select the best combinations of techniques in a **data-driven** way. The best ones may *include* a number of topological steps as part of a larger **ML pipeline**

Featurization: Produce "**features**" (scalars or vectors) which are amenable to processing by ML algorithms: **permutation-invariant functions**, **explicit vectorisations**, **learned representations**, …

Hyperparameters: Typically, several are involved within each choice of featurization technique (example: pixel size for persistence images)

Large-scale cross-validation routines: Must involve all hyperparameters and model choices at once, topological or not.

giotto-tda: Pillars

Giotto-tda

Seamless integration with widely used ML frameworks: inherit their strengths and allow for creation of heterogeneous ML pipelines. Python + *scikit-learn*

Code modularity: "Lego blocks" approach. Topological algorithms as *scikit-learn transformers*

User-friendliness and **familiarity** to the broad data science community

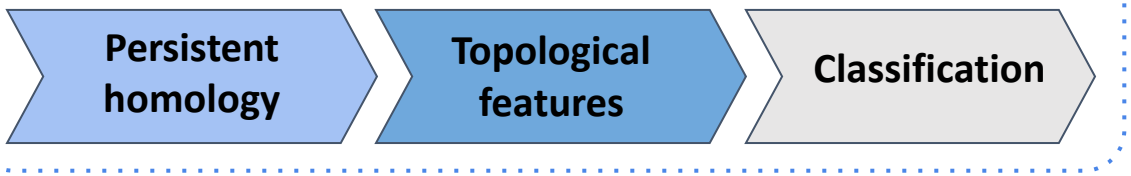
Standardisation: Allow for integration of most available TDA techniques into a generic framework

Performance: Parallelism and state-of-the-art C++ backends

Data structures: Support for **point clouds**, **time series**, **graphs** and **images**

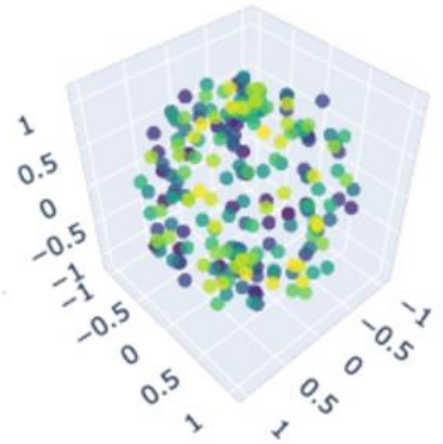
Persistent Homology

Pipeline



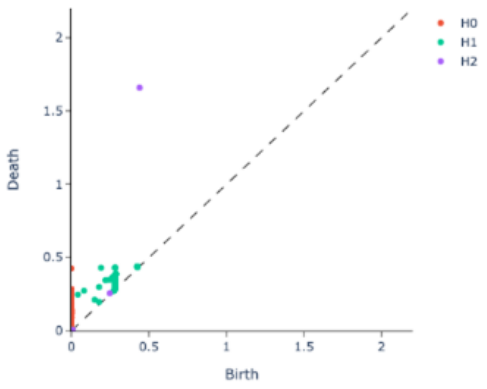

```
# Generate point clouds
clouds, labels = make_point_clouds(n_samples_per_shape=100,
                                   n_points=200,
                                   noise=0.2)

# Split between training set and test set
clouds_train, clouds_test, labels_train, labels_test = \
    train_test_split(clouds, labels)

# Define an end-to-end classification pipeline
persistence_pipeline = make_pipeline(
    VietorisRipsPersistence(),
    PersistenceEntropy(),
    RandomForestClassifier()
)

# Fit the whole pipeline on the training set
persistence_pipeline.fit(clouds_train, labels_train)

# Evaluate the model on the test set
persistence_pipeline.score(clouds_test, labels_test)
```

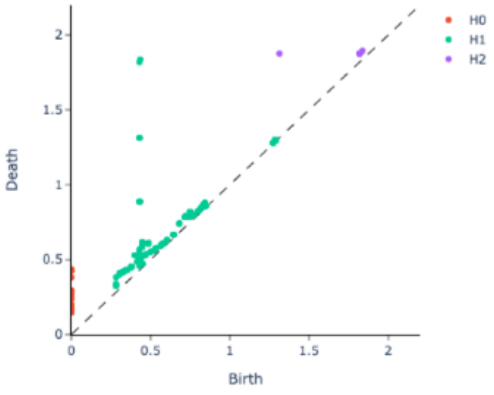
Sampled sphere



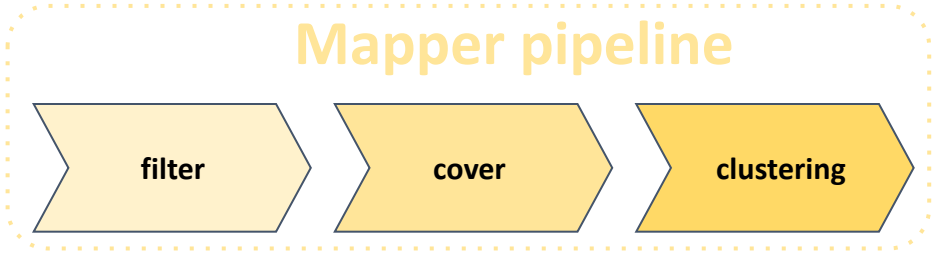
Sampled torus

Persistence diagrams





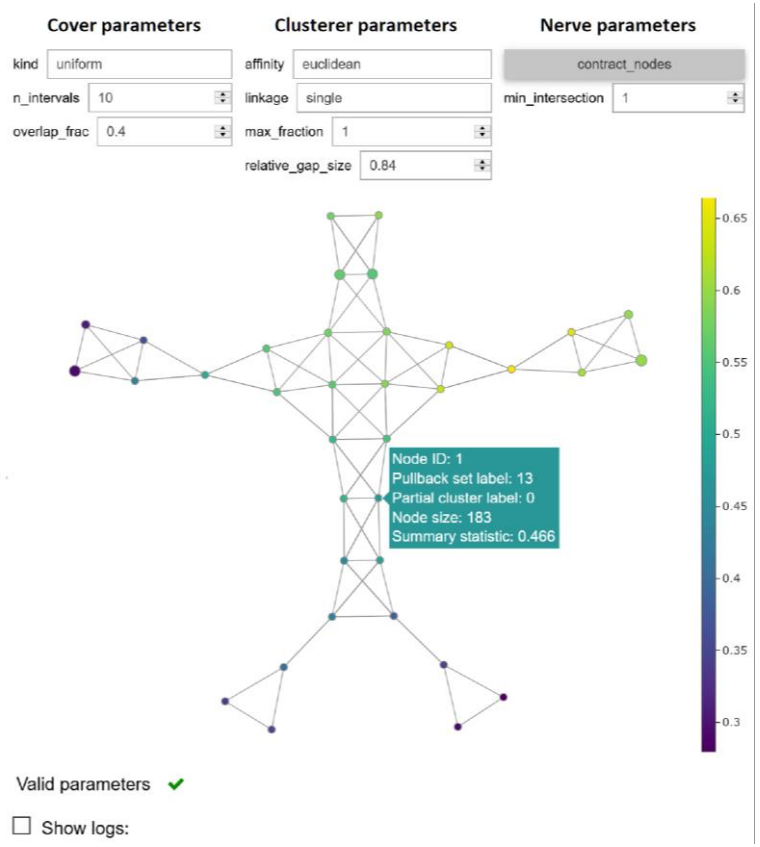
Mapper



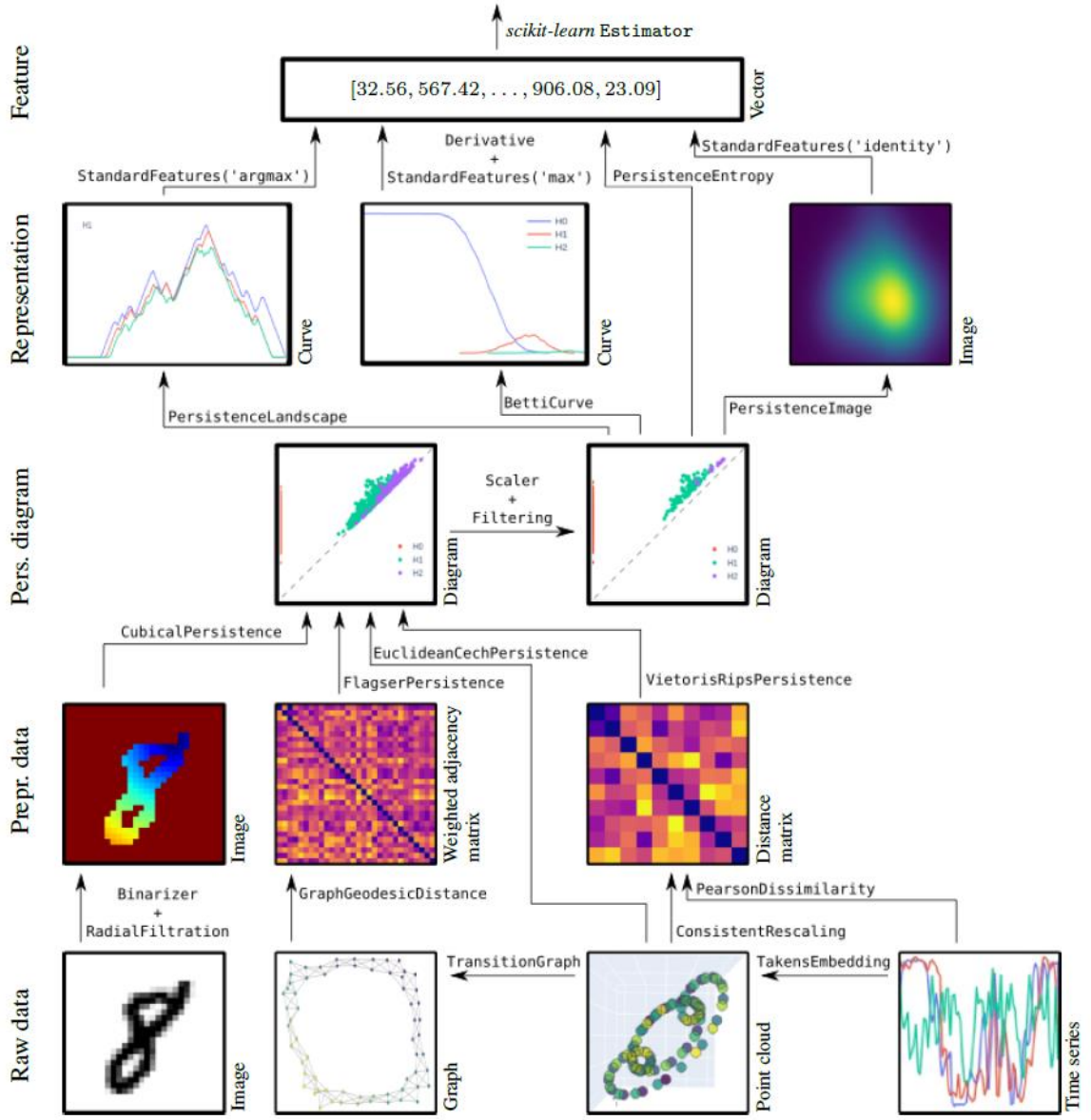

```
# Filter function can be any sklearn Transformer
filter_func = Projection(columns=[0, 1])
# Define cover
cover = CubicalCover()
# Choose clustering algorithm
clusterer = DBSCAN()

# Initialise pipeline
pipe = make_mapper_pipeline(
    filter_func=filter_func,
    cover=cover,
    clusterer=clusterer
)

# Generate interactive plot (Jupyter required)
plot_interactive_mapper_graph(pipe, alien)
```

Point cloud/metric space

Topological summary



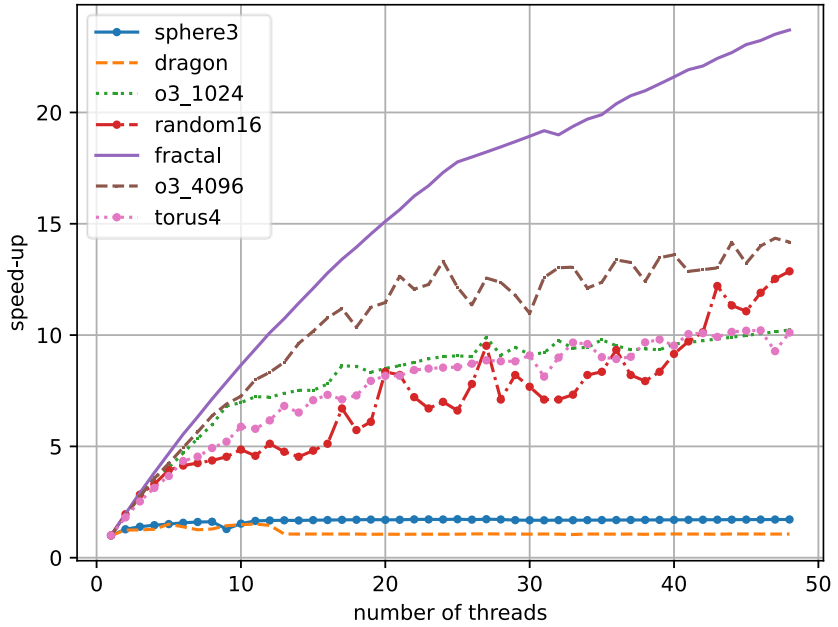
“Endless” possibilities



More Giotto topological tools

giotto-ph

- State-of-the-art, multicore computation of persistent homology of Vietoris—Rips filtrations
- Supported: **point clouds**, **graphs** with node and edge weights
- Makes fast backpropagation through persistent homology possible



pyflagser

- Fast persistent homology of directed flag complexes
- Like *giotto-ph*, but for **directed graphs**
- Example application: neuroscience

References

- G. Tauzin et al: ***giotto-tda*: A Topological Data Analysis Toolkit for Machine Learning and Data Exploration** (extended NeurIPS version, JMLR version)
- J. Burella Pérez et al: ***giotto-ph*: A Python Library for High-Performance Computation of Persistent Homology of Vietoris–Rips Filtrations** (arXiv:2107.05412)

Sources on GitHub: github.com/giotto-ai/{giotto-tda, giotto-ph, pyflagser}

Docs: giotto-ai.github.io/gtda-docs

Tutorials & examples: giotto-ai.github.io/gtda-docs/0.5.1/notebooks

API reference: giotto-ai.github.io/gtda-docs/0.5.1/modules

What would you like to do with *giotto-tda*?

Your help is welcome on GitHub: https://github.com/giotto-ai/giotto-tda

We are always **looking to integrate:**

- New algorithmic developments
- More preprocessing techniques
- More kernel and vectorization methods
- … And other important features!

Chat with us!

- Slack: https://slack.giotto.ai/
- GitHub discussions: https://github.com/giotto-ai/giotto-tda/discussions

Today’s tutorials

Navigate to https://github.com/ulupo/giotto-tda_demo and follow the installation instructions there

Interrupt & ask questions!