A Survey of Topological Data Analysis (TDA) Methods Implemented in Python

Jeffrey Ray and Marcello Trovati(✉)

Department of Computer Science, Edge Hill University, Ormskirk, UK
{rayj,trovatim}@edgehill.ac.uk

Abstract. The aim of this paper is to provide an overview of the current Python implementation to address a variety of topics in Topological Data Analysis, which include Persistent Homology, Manifold Learning and Mapper. We will discuss the effectiveness of each process based upon existing literature where TDA has been investigated. The purpose of this work is to inform future research efforts focusing on the implementation of TDA methods for managing and discovering, patterns, and trends for Big Data.

1 Introduction

The extraction of meaningful and actionable information from Big Data is at the core of much of the research in this field [1]. In fact, organisations and companies are likely to store enormous quantities of data, which are continuously produced by their external and internal processes. Understanding and assessing the value and relevance of actionable information contained within this wealth of data, is a key challenge.

Topological Data Analysis (TDA) is a new research area, which utilises topological concepts to classify and analyse data [2]. Broadly speaking, the focus of TDA is on the structure and *shape* of data, in terms of the interconnections between its components. One aspect of topology is the ability to classify objects based on their common properties, or in other words, those exhibiting invariant features. A example in topology, is that by stretching and deforming a doughnut, it can be turned into a coffee mug, as depicted in Fig. 1. In fact, there is a "hole" in both the handle of the mug and at the centre of the doughnut.

In many clustering and classification algorithms, the geometrical properties of the data investigated play an important role. In fact, many of such approaches focus on the concept of distance, which has to be defined according to the mathematical spaces where the data is embedded [3]. On the other hand, TDA focuses on the notion of shape and connectivity that such data exhibit. As in the example above, although a doughnut is clearly different from a coffee mug, they share the common property of having a single hole in them, which can be used to classify them in terms of their connected components [3]. In fact, no matter how much they are stretched, they will still be characterised by the existence of such a hole.

L. Barolli et al. (eds.), *Advances in Intelligent Networking and Collaborative Systems*,
Lecture Notes on Data Engineering and Communications Technologies 8,
DOI 10.1007/978-3-319-65636-6_54

Fig. 1. Transforming a doughnut into a coffee mug

Topology is usually specified by the metric of its corresponding space. Therefore, there is no strict requirement to limit to a specific coordinate system, thus allowing a more flexible approach [3].

A crucial aspect of TDA is the concept of *persistent homology*, which focuses on the identification of the topological properties which remain invariant [4]. Figure 2 depicts three separate datasets, which have in common the distinctive (topological) hole. Despite being different at a micro level, they could be regarded to be sufficiently similar. Persistent topology allows a classification of datasets based on such properties [5]. In particular, it is characterised by homology groups, so that two shapes are considered similar if they are isomorphic equivalent. In other words, datasets which can be topologically deformed and stretched into the same shape would fall into the same category. The three datasets in Fig. 2 could be intuitively interpreted as referring to a similar scenario, with some added noise. In fact, TDA allows noisy data to be addressed in a more efficient manner [2]. On the other hand, the two objects depicted in Fig. 3 are not topologically equivalent, as they have different connected components in terms of their number of holes.

One of the fundamental concepts of persistent topology, and homology in general, is simplicial complexes. These are space triangulations defined by combined, non-overlapping polyhedra, covering a topological space. A trivial, yet informative example of a space triangulation is image pixellation, where a real image is covered with pixels, to provide an accurate representation. One of the most important aspects of simplicial complexes is that fact that they provide an "approximation" of the object they are covering. Examples of triangulations include Voronoi diagrams, Delaunay triangulations, Vietoris and Čech complexes. Broadly speaking, they are defined by either a specific distance, or in terms of ball intersections whose centres are the data points. For more details, refer to [4]. Furthermore, the adjacency graphs generated by these triangulations can provide a variety of information on their invariant topological properties, and therefore relevant to TDA investigation [2].

The aim of this paper is to provide a survey of the Python implementations which can be used in TDA. The paper follows the following structure: in Sect. 2

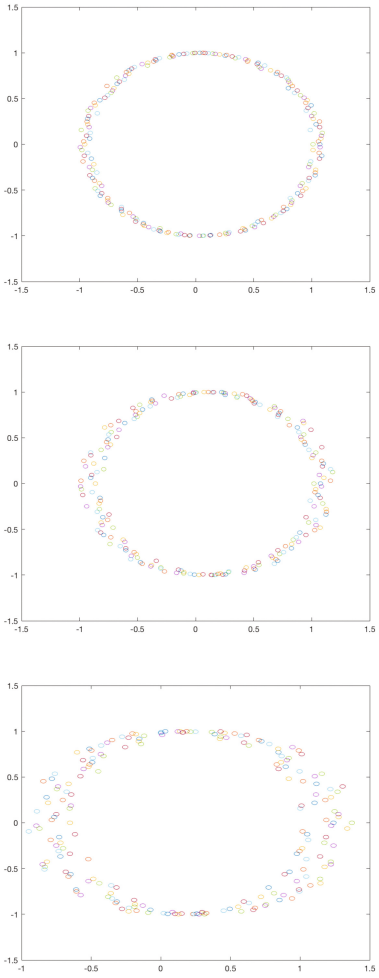

Fig. 2. Three datasets with similar topological properties.

Fig. 3. Two sets with different topological properties.

the existing technology is discussed, and Sect. 3 will discuss specific features of some Python methods and algorithms currently used in this research field. Section 4 will present the final remarks and future research.

2 State-of-the-Art TDA Methods

In [6], the authors discuss the application of the mapper TDA to noisy computer network data collected by a "darknet", with the goal of identifying activities such as port scanning and DDos attacks. In particular, they consider a dataset comprising over 3 million data points, which would prove difficult to visualise using traditional methods. Previously, software packages such as Suricata [8] would be utilised. However, it would not be sufficiently robust to successfully address noise whilst identifying all the attack patterns. The default Mapper code has been recently extended with KeplerMapper [9] for Python 3 (instead of Scikit-Learn [10]), and utilises C for the DBScan clustering to improve efficiency.

Motivated by the fact that large unstructured data sets can be visualised to provide valuable information, in [11] a commercial implementation based on TDA is described. Its objective is to provide a reliable diagnosis system to analyse large biomedical datasets, which has been shown to have high accuracy. In order to achieve this, TDA is used to define specific artificial neural networks coupled with Kolmogorov-Smirnov test [12]. The software package Ayasdi [7] is another commercial package to provide topological data analysis.

Biomedical data are often incomplete. Many analysis methods suffer when incomplete rows must be omitted, as valuable data might be subsequently removed. In comparison, the use of TDA does not require incomplete rows to be removed and can therefore produce information using all available information.

The use of TDA as a first stage in data analysis enables a wider range of initial data to be sampled, creating a more detailed grouping of data, regardless of any missing or incomplete fields. In fact, TDA allows for generality, as any notion of similarity can be successfully exploited in this context. On the other hand, classic machine learning algorithms typically require a comparably high level of similarity to produce any meaningful output.

2.1 Topological Algorithms

The methods discussed above fall into three main categories: persistent homology and Mapper and Manifold Learning, which will be discussed in this section, with particular focus on their Python implementation.

As described in Sect. 1, persistent homology is an algebraic method to assess topological features of shapes, via suitable triangulations, simplicial complex are defined in terms of a distance function of the underlying space [13]. This algorithm is relatively robust and well tested in the academic field [14]. An implementation of the persistent homology algorithm is found in the Python library Dionysus [15], which contains a variety of algorithms (Lower-Star, Vietoris-Rips, etc.) to cater for a large number of data shapes and types. Furthermore, Dionysus is a Python interpreter, which has been shown to be computationally efficient (but slower than a pure C++ implementation), whilst producing accurate data representation [15].

Manifold Learning, allows data of high dimensionality to be visualised by extracting key underlying parameters which form a low dimensional manifold.

Manifold Learning has various approaches, each implementing the extraction of data in a slightly different manner. The ability to utilise many of these methods can be achieved through the widely used and well document Python library Scikit-Learn [16]. This Python library allows Locally Linear Embedding, Modified Locally Linear embedding, Hessian Eigen mapping, Spectral Embedding, Local Tangent Space alignment, Multi-dimensional Scaling and $t-$distributed Stochastic Neighbour Embedding. With such a wide and carefully implemented set of Manifold Learning algorithms, it is possible to carry out TDA allowing for rapid and accurate data analysis of almost any data type.

Mapper is a widely utilised algorithm [17], which has also been successfully implemented in Python as Python Mapper [18]. The Mapper algorithm uses multiple filter functions to allow a coordinate system to represent a set of data, thus reducing complexity of the data set via dimensionality reduction [19]. Data points are connected via non-empty intersections between identified clusters, resulting in a topological summary of the data [20]. The Mapper algorithm can be successfully applied to cloud point data analysis, and it offers an alternative solution to the Dionysus implementation. However, the benefit of Mapper compared with Dionysus, is that its implementation is in pure Python as opposed to a port from C++. This creates a more efficient and more manageable code when used in a live environment.

3 Discussion

As discussed above, the Dionysus and Mapper algorithms focus on point cloud data sets, which implies that the data must be embedded onto a specific coordinate system. Mapper also supports two-dimensional and one-dimensional data sets defined as two-dimensional vector input data and one dimensional pairwise distances, respectively. This approach enables a more flexible and efficient method of analysing data.

Even though the Dionysus library also allows two-dimensional inputs, it introduces a limit in the construction of an alpha shape filtration [21]. As a consequence, from a data analysis point of view, the ease and flexibility of the Mapper algorithm and Python Mapper solution is preferable to the Dionysius library.

The Manifold Learning algorithms contained within the Scikit-Learn package require the data to be embedded onto a low dimensional sub-manifold, as opposed to the Mapper algorithm, which allows higher dimensionality. In particular, they need the dataset to be locally uniform and smooth often with restriction on sampling uniformity. In contrast, the Mapper algorithm output is not intended to faithfully reconstruct the data or reform the data to suit a data model, as it provides a representation of the data structure.

The Manifold learning and Mapper solutions provide a useful set of data analysis tools, which enable a suitable representation of data structures. Furthermore, they provide different methods based on specific assumptions on data inputs and output. Therefore, a choice between them can only be made once the corresponding data set has been created and its structure is known. The ease

of implementation due to both libraries being native to the Python programming language, allows an integration with other popular data science Python packages. This clearly facilitates the creation of an efficient and computationally feasible data analysis platform.

The Mapper algorithm has been extensively used for commercial data analysis tools, due to its ability to process complex data sets containing over 500,000 features. This also enables complex Big Data to be handled without the requirements and complexity of deploying Hadoop, map reduce and SQL data base, prove the flexibility and reliability of the algorithm. Figure 4 shows details of the performance of the above algorithms.

Method	Properties	Limitations	Strengths
Python Mapper	Native Python Combines Filter functions, the Mapper algorithm and visualisation results	A pure python implementation, slow than a traditional programming language implementation such as C	Robust algorithm capable of relative equal output regardless of sample size. Easy integration with Other Native python libraries
Dionysus (Persistent Homology)	C++, with Python bindings Persistent homology computation, Vineyards Persistent cohomology computation, Zigzag persistent homology.	Difficult installation, requires a large list of dependencies Not all functionality is completely available in Python	Highly resistant to data noise.
SciKit Learn (Manifold Learning)	Native Python. Non-linear dimensionality reduction. Wide variety of possible approaches.	Same scale must be used over all features - Manifold learning uses nearest-neighbour search. Certain input configurations can lead to singular weight matrices (more than two points identical)	Able to adapt Linear Frameworks to be sensitive to non-linear structures in data. Comprehensive community support Integrates seamlessly with widely used Machine learning libraries already available in Python

Fig. 4. Performance data for networkX and Graph tools.

4 Conclusion

The manifold learning and Mapper methods for TDA allow greater flexibility and data input sources compared to the Dionysus implementation. Furthermore, highly specialised data scientists have been contributing to the Scikit-Learn and Mapper projects, creating a peer reviewed and well maintained implementation, which is not the case for Dionysus. Industry Standard analysis tools, such as Ayasdi, have been built with the Mapper algorithm at their core, demonstrating their industry appeal and credibility. This suggests that data analysis platforms

aiming to obtain efficiently interpreted results with applications to a decision-making process should prioritise the implementation of the Mapper algorithm.

References

1. Wang, L., Wang, G., Alexander, C.A.: Big data and visualization: methods, challenges and technology progress. Digit. Technol. Digit. Technol. **1**(1), 33–38 (2015)
2. Carlsson, G., Harer, J.: Topology and data. Bull. Math. Soc. **46**(2), 255–308 (2009)
3. Janich, K.: Topology. Springer, New York (1984)
4. Edelsbrunner, H., Harer, J.: Computational Topology: An Introduction. American Mathematical Society, New York (2010)
5. Weinberger, S.: What is persistent homology? Not. AMS **58**(1), 36–39 (2011)
6. Coudriau, M., Lahmadi, A., Francois, J.: Topological analysis and visualisation of network monitoring data: darknet case study. In: 2016 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1–6 (2016)
7. Ayasdi (2017). https://www.ayasdi.com. Accessed 24 May 2017
8. Suricata (2017). https://suricata-ids.org/. Accessed 24 May 2017
9. MLWave: kepler-mapper: KeplerMapper is a Python Class for Visualization of High-Dimensional Data and 3-D Point Cloud Data. kepler-mapper (2017)
10. Scikit-learn: machine learning in Python scikit-learn 0.18.1 documentation (2017)
11. Rucco, M., Falsetti, L., Herman, D., Petrossian, T., Merelli, E., Nitti, C., Salvi, A.: Using topological data analysis for diagnosis pulmonary embolism. Phys. Q-Bio (2014). arXiv14095020
12. Upton, G., Cook, I.: Kolmogorov-Smirnov test - Oxford Reference. http://www.oxfordreference.com.edgehill.idm.oclc.org/view/10.1093/acref/9780199679188.001.0001/acref-9780199679188-e-868. Accessed 24 May 2017
13. Goodman, J.E.: Surveys on Discrete and Computational Geometry: Twenty Years Later: AMS-IMS-SIAM Joint Summer Research Conference. American Mathematical Society (2008)
14. Edelsbrunner, H., Letscher, D., Zomorodian, A.: Topological persistence and simplification. In: Proceedings 41st Annual Symposium on Foundations of Computer Science, pp. 454–463 (2000)
15. Morozov, D.: Welcome to Dionysus documentation. http://www.mrzv.org/software/dionysus/. Accessed 29 May 2017
16. Scikit-Learn: Manifold learning scikit-learn 0.18.1 documentation. http://scikit-learn.org/stable/modules/manifold.html. Accessed 29 May 2017
17. Singh, G., Memoli, F., Carlsson, G.: Mapper: a topological mapping tool for point cloud data. In: Eurographics Symposium on Point-Based Graphics (1991)
18. Müllner, D., Babu, A.: Python Mapper: an open-source toolchain for data exploration, analysis, and visualisation. Stanf, Edumuellnermapper (2013)
19. Carlsson, G.: Topology and data. Bull. Am. Math. Soc. **46**(2), 255–308 (2009)
20. Chow, Y.Y.: Application of Data Analytics to Cyber Forensic Data A Major Qualifying Project Report. MITRE Corporation, McLean (2016)
21. Giesen, J., Cazals, F., Pauly, M., Zomorodian, A.: The conformal alpha shape filtration. Vis. Comput. **22**(8), 531–540 (2006)