Can we use Topological Data Analysis to learn about Image Data?

Shanik Dassenaike

April 19, 2018

id15663@my.bristol.ac.uk

Abstract

Can we use topological data analysis to learn about image data?

Topological data analysis is the method of transforming data of some description into a topological shape to analyse it. Its key insight is that data fundamentally has shape, and that shape matters. It can be used to find out information about the data that is obscured from other methods of analysis, for example hidden patterns. Moreover, a lot of data is very high dimensional, and as such is very sparse. Traditional analysis techniques rely heavily on dimensionality reduction and chosen metrics in order to say anything useful about the data, and can be sensitive to errors caused by noise. With topological data analysis, dimensionality reduction is effectively built-in, and it is insensitive to the particular metric used. It is also robust to noise, as again it is the overall shape of the data that is important.

Image data is often very high dimensional, considering that it is usually made of $n \times n$ (with n typically $\gg 10^2$) pixels in 3 dimensions, sometimes with a fourth alpha dimension also. We posit that topological data analysis provides a novel way to consider this data, and learn from it. There is clear intuition in this; images are fundamentally made up of different (geometric) shapes and there is no reason this should not translate well into topology. Additionally, the high dimensionality of image data is often a problem for traditional techniques - most of the dimensions provide no useful information and do not relate to the mapping or properties of the image itself.

Research about/Mathematical Foundations of Topology

The below uses:

- ftp://ftp.mpi-sb.mpg.de/pub/conferences/adfocs-07/graphics/zomorodian-slides-1.pdf
- https://www.researchgate.net/publication/310406546_Persistent_homology_a_step-by-step_introduction_for_newcomers
- https://math.stackexchange.com/questions/884666/what-are-differences-between-affine-space-an
- https://en.wikipedia.org/wiki/Affine_space
- https://en.wikipedia.org/wiki/Simplex
- https://en.wikipedia.org/wiki/Simplicial_complex
- http://mathworld.wolfram.com/AffineSpace.html

Although the results and analysis contained in this thesis do not require a deep understanding of topology or image processing to interpret, herein we provide a mathematical basis and low-level explanation of some of the image processing algorithms used, to aid in understanding the work to follow and the conclusions that can be drawn from the results. To avoid over-explaining, however, we assume the reader has knowledge of basic Linear Algebra and Group Theory.

Affine Spaces

Affine Spaces are structure that generalise vector spaces, avoiding concepts of distance and measure of angles; the only properties that remain are related to parallelism and ratio of lengths of parallel line segments.
Suppose we had a vector space V over a field $\mathbb{F}$. Let A be a non-empty set. For any vector $\mathbf{v} \in V$ and element $p \in A$, we define addition $p + \mathbf{v} \in A$ with the following conditions:

- $(p + \mathbf{v}) + \mathbf{w} = p + (\mathbf{v} + \mathbf{w})$
- $\forall q \in A, \exists! \mathbf{w} \in V$ s.t. $q = p + \mathbf{w}$
- $p + \mathbf{0} = p$ (Note this is implied by the above)

Then A is an affine space, and $\mathbb{F}$ is called the coefficient field.

An intuitive understanding of affine spaces is to consider them as vector spaces in which we have 'forgotten' where the origin is. As such, no vector has a unique origin and thus cannot be associated with any particular point.
An example of an affine space is the plane in $\mathbb{R}^3$ defined by $< x, y, 1 >$, i.e. the xy-plane sitting at $z = 1$; this is clearly not a vector space as it does not contain the origin. However, we might still subtract two points in this plane and obtain a vector, just as in a vector space. On the other hand, in this space we cannot just take a point and find a vector with it, as there is no origin to define this vector from; similarly, we cannot add two points together to obtain a vector as they are not measured relative to an origin.
Many notions carry over from vector spaces to affine spaces in some way; suppose we have an affine space A and a subset $X \subseteq A$. Then:

- The smallest affine subspace containing X is called the affine span of X
- X is affinely independent if the affine span of any proper subset of X is a proper subset of the affine span of X.

Simplicial complexes

Discretised objects, such as mathematical graphs, or digital images, can be represented by simplicial complexes. These are collections of "well-glued" bricks called simplices.

Simplex

A simplex generalises the notion of a triangle or tetrahedron to arbitrary dimensions. Formally, suppose we had $k + 1$ affinely independent points, $u_0, u_1, ..., u_k \in \mathbb{R}^k$. The convex hull of these points is the k-simplex σ they determine; it is the set of points

$$C = \left\{ \theta_0 u_0 + ... + \theta_k u_k | \sum_{i=0}^{k} \theta_i = 1 \text{ and } \theta_i \geq 0 \ \forall i \right\}$$

We have that a 0-simplex is a point, a 1-simplex an edge, a 2-simplex a triangle, a 3-simplex is a tetrahedron and the notion generalises on. The dimension of σ, $dim(\sigma) = k$.
For a simplex σ, any non-empty subset of the points generating σ whose convex hull itself is a simplex is called a face of σ.

Simplicial Complex

We can "glue together" simplices to form a simplicial complex.
A simplicial complex Σ is a finite set of simplices that satisfies the following "gluing" conditions:

- Any face of a simplex in Σ is itself in Σ
- The intersection of any two simplices $\sigma_1, \sigma_2 \in \Sigma$ is a face of both σ_1 and σ_2

The dimension of a simplicial complex Σ, $dim(\Sigma)$, is the largest dimension of its simplices.

Topological Spaces

- We have a set of points, X
- An open set is a subset of X.
- A Topology on X is then a set of open sets $T \subset 2^X$, such that:
  1. If $S_1, S_2 \in T$, then $S_1 \cap S_2 \in T$
  2. If $\forall j \in J$, we have that $S_j \in T$, then $\bigcup_{j \in J} S_j \in T$
  3. $\varnothing, X \in T$
- $\mathbb{X} = (X, T)$ is a topological space
- Different topologies are possible
- A Metric Space is an open set defined by some metric

Note that the set X on its own may be called a topological space.

Cover

- We have a set of points, X
- If C is a family of sets, and the union of C contains X, then C is a cover of X

This can be applied specifically to topological spaces:

- Let X be a topological space.
- Suppose $C = \{U_i | U_i \subset X \text{ for } i \in I\}$ is a family of subsets of X (with I some index set)
- Then C is a cover of X if the union of C is equal to the whole set X, or in other words if $\bigcup_{i \in I} X_i = X$
- We then say that C covers X, or that the sets X_i cover X.

Notice the difference in the definition is that in topology, the family of sets C must consist of subsets of X, and we see that when this is applied to the earlier definition, it coincides with that given for topological spaces - as the union of the subsets necessarily cannot be any larger than the whole set X.

What is a Homeomorphism?

Suppose we have topological spaces $\mathbb{X}, \mathbb{Y}$, with a continuous bijective map $f : \mathbb{X} \to \mathbb{Y}$ (with f^{-1} also continuous). Then:

- f is a homeomorphism
- $\mathbb{X}$ is homeomorphic to $\mathbb{Y}$
- $\mathbb{X} \approx \mathbb{Y}$
- $\mathbb{X}$ and $\mathbb{Y}$ have the same topological type

Computational Topology

Our input is often Point Cloud Data, which is:

- Massive
- Discrete
- Nonuniformly Sampled
- Noisy
- Embedded in $\mathbb{R}^d$, sometimes with $d \gg 3$

We want to find out its shape.

Image Processing

Many aspects of computation, for example computer vision, rely on image processing algorithms to work with images and image data. One reason for this that is common with the requirement to use it with TDA is dimensionality reduction and, more generally, reducing the amount of data required to work with - often due to computational constraints.

Two well-known image processing algorithms are the scale-invariant feature transform (SIFT) and the speeded-up robust featues (SURF) algorithms. These are both proprietary algorithms, so we can use them for the academic purposes of this paper, though cannot explain them in great detail.

Both algorithms work by creating 'key-points', and associate a 'descriptor' to each. What this means is that in any image, there are points which may be considered 'interesting'. The algorithms work by finding sets of interesting points, each of which is then described mathematically with a vector in a relatively large number of dimensions (typically 64 for SURF and 128 for SIFT). When all key-points with their descriptors are considered together for an image, a feature description of the image is obtained.

It is important to note that the key-points say little about the image by themselves; each key-point is generally only a position within the image and an area around that position (different algorithms may include more or less data than this), but it is not enough to describe the image fully. Key-points are, as their name implies, little more than just points in the image. The descriptors are what tell us what exactly is 'interesting' about each key-point, or in other words, the descriptors are what provide the numerical description of each key-point. As such, while key-points may change from image to image, if they are describing the same object, their descriptors should have little distance between them - recall that mathematically descriptors are vectors, so we can compare the distance between any two descriptors - or potentially even be the same, and so descriptors are used to compare key-points (this is important to note for our use). This is only dependent on the algorithm in the sense that descriptors are not unilaterally invariant to transformations, but this issue is part of what SIFT and SURF seek to solve.

There are papers that show that an image can, to some extent, be reconstructed from these feature descriptions. Moreover, these algorithms are often used by extracting features from some training data, and then using this to recognise features in other images - i.e. these algorithms are designed to be able to recognise objects across images. For our purposes, however, it suffices to see that this means the features serve as a compressed representation of the images, retaining its interesting features, which will ultimately be from what we extract our topological information.

SIFT

The SIFT algorithm was patented in 1999, creating a set of key-points with associated descriptors. These descriptors are typically 128-dimensional, and are scale-invariant. So if two images both contain the same object, but one is more zoomed-in such that the object appears larger in the image, the matching key-points for the object will also contain the same descriptors. SIFT descriptors are also invariant to orientation, illumination changes, and partially invariant to affine distortion (i.e. transformations in affine space).

To find the key-points and descriptors, SIFT creates a scale space, and then uses this to approximate the Laplacian of Gaussians (using difference of Gaussians from those used to contstrcut the scale space). Maxima and minima in the difference of Gaussian provide key-points, and some of these are then discarded (if they are edges or low-contrast points). Each key-point is then given an orientation, and finally descriptors are generated for each key-point based on the area around it.

SURF

The SURF algorithm was patented in 2006, inspired by SIFT but intended to be faster and invariant under more image transformations. Although SIFT descriptors are orientation-invariant, they are not fully robust with respect to rotations, where SURF is.

The outline of generating SURF key-points and descriptors is largely the same as SIFT, but SURF does not use Gaussians to approximate the Laplacian of Gaussians in the way that SIFT does, instead approximating them with integral images which speeds up computation. The computation of the descriptors uses the integral images also, saving on computation again.

Research about TDA itself

the below uses:

- https://arxiv.org/pdf/1710.04019.pdf
- https://www.youtube.com/watch?v=XfWibrh6stw
- https://www.youtube.com/watch?v=fUvl-B2lx5Q
- https://www.nature.com/articles/srep01236

The pipeline of TDA

Most existing TDA methods use the following pipeline as a basis. This section is completely unoriginal, is there way to digest and rewrite it in a way that describes how I use TDA?

1. assume the input is a finite set of points, with a notion of distance/similarity between them. This distance could be induced by the metric in the ambient space (e.g. the Euclidean metric, when data is embedded in $\mathbb{R}^d$), or it can come as an intrinsic metric defined by a pairwise distance matrix. The definition of this metric is usually given as an input or guided by the application - we notice that the choice of metric might be critical to revealing interesting features.

2. To highlight the topology underpinning the data, we build a "continuous" shape on top of it. Oftentimes this is a simplicial complex or a nested family of simiplicial complexes - a filtration - reflecting how the data is structured at different scales. We can see simplicial complexes as high-dimensional generalisations of neighbouring graphs that are classically built on top of data, in many standard data analysus or learning algorithms. We want to be able to define such structures that are proven to reflect relevant information about the structure of data, and that can be effectively constructed and manipulated in practice.

3. Topological or geometric information is extracted from the structures built on top of the data. We can achieve two results from here; we could get a full reconstruction - typically a triangulation - of the shape underlying the data, and from this topological/geometric features can be easily extracted. On the other hand, we can get crude summaries or approximations, and from these the extraction of relevant information requires specific methods, for example persistent homology. Further to identifying interesting topological/geometric information, and visualising and interpreting this, the challenge at this step is to show its relevance, in particular its stability with respect to perturbations or presence of noise in the input data. For that purpose, understanding the statistical behavior of the inferred features is also an important question.

4. The extracted topological and geometric information provides new families of features and descriptors of the data. We use these to better understand the data, particularly through visualization, but was also can be combine these families with other kinds of features for further analysis and machine learning tasks. Showing the added-value and the complementarity (with respect to other features) of the information provided by TDA tools is an important question at this step.

There are three key concepts in topology that lend power to analysing or understanding data by find patterns in its shape: Coordinate Invariance, Deformation Invariance, and Compressed Representation.

1. **Coordinate Invariance:** Topology studies shapes in a coordinate-free manner; what that means is that the coordinate system in which we view a shape does not affect the properties of it that we study. The topological constructions depend only on the distance function intrinsic to the metric space within which the shape is specified.

2. **Deformation Invariance:** The properties studied in topology are also invariant under "small" deformations - despite any "stretching" or "squashing", as long as the shape is not "torn" or "reglued", any property topology studies is unchanging. For example, if we were

to write down the capital letter "A" on an elastic surface, and then stretched it in some directions, it will still retain its closed triangle and two legs pointing out. Similarly, a capital "A" written in different fonts is still clearly a letter A (and in fact humans are particularly good at recognising deformation-invariant properties), as the fundamental parts of the letter haven't changed.

3. **Compressed Representations:** oftentimes an object that we want to study is very highly (potentially infinitely) complex in its detail and information contained. Topology allows us to approximate the object with a finite representation - a triangulation. This may mean identifying the object with a simplicial complex or a network, for example identifying a sphere with an icosahedron or a circle with a hexagon. In either case we go from infinite points on a surface to a small, finite number of points, edges and faces, so we lose some information (with these approximations curvature is an example) but we retain the important topological feature e.g. a loop

An understanding of both the pipeline outlined earlier as well as these concepts is key to analysing the results we achieve. Our realisation of the pipeline will involve pre-processing the data-sets (images in our case) to find compressed representations, applying a filter function to create simplicial complexes, applying an algorithm to extract the topological data and then analysing these results. If, for example, we see that the results for both data-sets consists of two loops attached, with two branches at either end of each loop, we can conclude that these images have some similar underlying structure that the topology is highlighting. Even if the results aren't exactly similar, as long as these topological features are consistent from the results of one data-set to another, the deformation and coordinate invariance allow us to consider the results to be the same (or at least fundamentally similar). This provides us with a basis for confirming the usefulness of TDA with respect to image data, or to show unexpected similarities between images.

Random cubical complexes model noise in digital images

Rough plan: Take an image; process it to get vectorial/clustered data. Produce a topological analysis of this data (e.g. create a simplicial complex over the clusters and find out how it looks/if there are any obvious properties). Jumble the data somehow (mix up the vectors while maintaining their distance metric), and then apply the same topological analysis to it, compare the outputs. Can we use some finite/easy amount of applications to achieve showing them the same

Method

Python Mapper

Python Mapper is a toolchain created by Daniel Mullner and Aravindakshan Babu. It realises the processing chain, at the least, running a data set through a filter function, then applying the Mapper algorithm created by Gurjeet Singh, Facundo MÃľmoli and Gunnar Carlsson, and finally visualising these results; these three steps together constitute a topological data analysis tool that we can use on an appropriate data-set. Indeed, we notice how this toolchain parallels the pipeline outlined earlier, other than the first step. The tool is presented as a GUI and also as a python module. This paper will demonstrate the application of both implementations to image data(, as well as explaining a new GUI catered towards analysing image data via Python Mapper).

Mapper algorithm

Gurjeet Singh, Facundo MÃľmoli and Gunnar Carlsson developed the mapper algorithm, and subsequently founded the company Ayasdi based on it. It is simple at its core: bin the data into overlapping bins, cluster each bin, and finally create a graph where vertices represent clusters, and two vertices are connected by an edge if they have common points. The assumptions made are that we have a point cloud with N points $x \in X$, and along with there is a filter function that maps the point cloud to the reals (or potentially $\mathbb{R}^2$, or the unit circle etc), $f : X \to \mathbb{R}$, for which the value is known for each of the N data points. It is also assumed that the inter-point distances between points can be calculated - this again lines up with our concept that distances are what are important in topology (and in fact can be key to understanding how data is structured). A covering of the data is found by taking the range of f over X, and dividing it into a set of

overlapping intervals (bins), S. For each interval $I_j \in S$, the set of points in X that map to points in I is determined, and the set X_j containing each of these sets clearly forms a cover. Thus we have constructed our 'overlapping bins'. So the next step is to cluster each bin.