Towards topological analysis of high-dimensional feature spaces.

Hubert Wagner and Paweł Dłotko

January 7, 2014

Abstract

In this paper we present ideas from computational topology, applicable in analysis of point cloud data. In particular, the point cloud can represent a feature space of a collection of objects such as images or text documents. Computing persistence homology reveals the global structure of similarities between the data. Furthermore, we argue that it is essential to incorporate higher-degree relationships between objects. Finally, we show that new computational topology algorithms expose much better practical performance compared to standard techniques.

1 Motivation

The purpose of this paper is to introduce concepts and techniques from computational topology in the context of image understanding and pattern recognition. We think that using such methods in conjunction with the standard tools present in these fields, can give rise to new effective solutions.

Faced with a large collection of documents, including images, it is useful to have a global view of this dataset. In this paper we argue that tools of computational topology can be used to capture topological structure of point-cloud data and that this information can be useful. In particular using persistent homology we are able to capture global, higher-dimensional patterns within a feature space.

Data mining methods often use graph-theoretical approaches [13]. Analysing the connected components of the graph of *similarities* between pairs of objects is a simple example. From a topological perspective, such analysis operates on 1-dimensional *complexes* (only pairs of documents are considered) and gives 0-dimensional topological information.

Higher dimensional relationships, i.e. relationships between larger subsets of data, are sometimes used in data-mining. For example, the number of triangles (3-cliques) is an important descriptor of the connectivity of a social or collaborative network [8]. Rather than finding just the *number* of

such higher-dimensional elements, we would like to compute their topological structure.

We believe that mining a higher dimensional *topological structure* within a set of objects can give an important insight into the data. For example, [2] shows that data coming from natural images form a topological Klein bottle.

The original motivation for our project was an application from the area of text-mining, as described in [19]. The following paper serves as an extension, putting these techniques in a wider context. To be specific, we show the applicability of this framework in the context of computer vision and image understanding. Additionally, we update some of the information contained in the previous paper [19], based on recently gained experience.

We intend to accomplish three goals. First, introducing a higher-dimensional similarity measure, we show that the point-cloud can be interpreted as a simplicial complex, with meaningful filtration values defined on simplices of all dimensions. Such a representation allows for treatment with topological tools. Second, we argue that a higher-dimensional analysis, based on topological tools, can indeed be interesting and relevant. Third, we demonstrate that using recent algorithmic techniques, much larger datasets can now be handled.

2 Input data

By a *feature space* we mean a vector space, where each coordinate corresponds to the numerical value of a certain feature. Each possible tuple of features' values can be represented as a point (vector) in this space. These points may correspond to images, text documents etc. depending on the application.

In general, our input consists of a set of objects (images, text documents etc.). We also choose a certain *similarity measure*, quantifying how related, or similar, objects are. Normally, similarity is a pairwise function. In our methodology, we consider higher-dimensional relationships, that is the relationships within sets of arbitrary size. In contrast, standard, graph-theoretical methods capture only pair-wise relationships, effectively operating on a *one-dimensional* structure.

The values of similarity range from 0 (completely unrelated objects) to 1 (indistinguishable objects). Therefore, in our method we identify objects with similarity equal to 1.

Importantly, we also define dissimilarity, $dsim(.) = 1 - sim(.)$, where sim is a similarity measure. Dissimilarity fits conveniently with the framework of persistent homology. Note that dissimilarity it is not necessarily a metric, but we do require $dsim(x, x) \leq dsim(x, y) = dsim(y, x)$.

3 Computational topology

In this section we give a brief introduction to computational topology. For a formal introduction see [6]. A paper by Carlson [2] is an important work, which shows that analysis of higher-dimensional data can be meaningful. A number of papers dealing with topological analysis in lower-dimensional spaces exist, but these techniques are hard or impossible to generalize to higher dimensions [14]. A recent paper by Zomorodian [18] deals with building Rips complexes of high dimensional data.

A finite collection of finite sets, S, is an abstract *simplicial complex* if for every $t \in S$ and for every $s \subset t$ we have $s \in S$. Every element $t \in S$ is a *simplex* and its *dimension* is defined as $card(t) - 1$. By S_k we denote the k-skeleton of complex S, i.e. all simplices in S with dimension $\leq k$. If $s \subset t$ and $card(t) - card(s) = 1$, we say that s is a *face* of t and t is a co-face of s. *(Co-)boundary* is the set of all (co-)faces of a simplex. A simplex of dimension 0,1,2,3 is respectively: *a vertex, an edge, a triangle* and *a tetrahedron*.

We outline the computations performed: Starting from the point-cloud equipped with a *dissimilarity measure* we construct a *filtered simplicial complex*, which encodes higher dimensional topological information, and can be viewed as a higher-dimensional analog of a graph. Then, we compute the *persistence diagram* which encapsulates *persistent homology* on this data. We now proceed to define and explain these concepts.

3.1 Čech and Rips complexes

A point cloud, can be imagined as a sample of an underlying space. We reconstruct this space as a union of balls of a certain radius. (Later we will use persistent homology, so instead of fixing this radius, it will become a parameter). There are two standard constructions, which yield a combinatorial representation of such a union of balls, in the form of a simplicial complex. Let $B_x(r)$ denote a ball centered at x with radius r. For a given point cloud P, we define the Čech complex:

$$\check{\text{C}}\text{ech(r)} := \{\sigma \subseteq P \mid \bigcap_{x \in \sigma} B_x(r) \neq \emptyset\}$$

Similarly, we define the Rips complex:

$$Rips(r) := \{\sigma \subseteq P \mid max_{a,b \in \sigma} d(a,b) < 2r\}$$

For sufficiently nice spaces, such as R^n with the standard topology, Čech complex has the homotopy type of the union of balls. In particular, it has the same homology [6]. Rips complex, being easier to compute, is usually used in practice, even though it might include some spurious topological information. In our setting the space might be more exotic, depending on

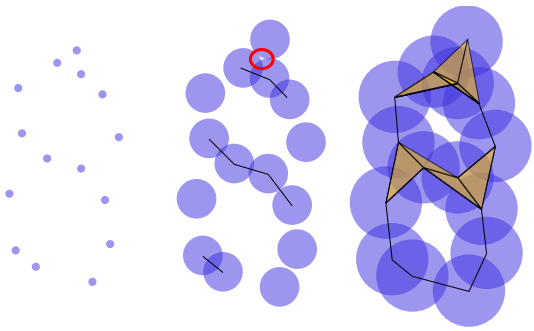

Figure 1: Example of a point cloud, and a union of balls of growing size. A simplicial complex, called Rips complex is overlaid, being a combinatorial representation of the union of balls. Persistent homology captures the changes in homology for the growing radius of balls.

the chosen dissimilarity measure, and in general the Čech property might not hold. Still, this is a useful intuition.

3.2 Persistent homology

Homology is a mathematical formalism used to define and identify basic topological features, called *holes*. Holes are defined for arbitrary dimensions, and in three ambient dimensions they are intuitive: 0-dimensional holes are related to the gaps between connected components, 1-dimensional ones can be viewed as tunnels (like a hole in a donut). 2-dimensional holes are cavities (inside of a balloon). See [6] for a formal definition of homology. By *homology class*, we simply mean an individual hole.

Persistent homology describes the changes in homology when a certain scale parameter is varied. This way it can be viewed as a multi-scale view of topology. More formally, given a simplicial complex $\mathcal{K}$ and a filtering function $g : \mathcal{K} \to [0, 1]$, *persistent* homology studies homological changes of the sub-level complexes: $\mathcal{K}_t = g^{-1}([0, t])$. Changing t from 0 to 1 induces a sequence of complexes called a filtration. The complex with the filtering function is called a filtered complex. Importantly, we require that $g(a) \leq g(B)$, whenever a is a face of B, which we call the *filtration property*. It implies that for every $t \in \mathbb{R}$, $\mathcal{K}_t$ is a *complex*, namely a simplex appears no sooner than its faces in a filtration.

Persistent homology captures the birth and death times of homology classes of the sub-level complexes, as t changes from 0 to 1. By birth, we mean that a homology class is created; by death, we mean it either becomes trivial or becomes identical to some other class born earlier. The *persistence*, or lifetime of a class, is the difference between the death and birth times.

Figure 1 shows three levels of a filtration, build over a point cloud simpled from a figure eight. At consecutive filtration levels, holes are created and then killed. We start from a number of connected components. In the middle level, the connected components merge, and a small 1-dimensional cycle is formed. This cycle (outlined by the red ellipse), however has small *persistence*, as it will immediately be glued-in as the balls grow. Then two

1-dimensional cycles are created. If we decreased the parameter further, the cycles would also be glued-in, but they persisted over a large change of parameter. We can identify three homology classes with relatively high persistence, namely the unique connected component, which persists forever, and the two one-dimensional cycles. Clearly, these homological features correspond to the apparent shape of figure eight.

An important justification for the usage of persistence is the stability theorem. Cohen-Steiner et al. [5] proved that putting some mild assumptions on two filtering functions f and g from $\mathcal{K}$ to $[0, 1]$, the so-called bottleneck distance (d_B, see [5]) between the persistence of $\mathcal{K}$ filtered by function f (denoted as $H^p(\mathcal{K}, f)$) and the persistence of $\mathcal{K}$ filtered by g ($H^p(\mathcal{K}, g)$) is upper bounded by the L_∞ norm of the difference between f and g:

$$d_B(H^p(\mathcal{K}, f), H^p(\mathcal{K}, g)) \leq \|f - g\|_\infty := \max_{x \in \mathcal{K}} |f(x) - g(x)|. \quad (1)$$

This result was generalized by Chazal et al. [4] to complexes arising from spaces which are equipped with a distance-like measure failing to satisfy the triangle inequality. The result holds, if $d(x, x) \leq d(x, y) = d(y, x)$, which is the case in our setting. Intuitively, if the input data is perturbed slightly, its persistence also changes a little. This property makes persistence robust, in particular against noise.

4 Interpreting input data

Given a point cloud and a pair-wise dissimilarity measure, one can build a Rips complex, filtered by the dissimilarity, as in [19]. In this case a simplex appears in the filtration, whenever all of its faces are present. Let us consider what this means for data interpretation: Effectively the dissimilarity between subsets of objects is defined as the *maximum* of the pair-wise similarities. It is not hard to imagine a situation with three pair-wise similar objects, which are not triple-wise similar, as shown in Figure 2.

4.1 Multidimensional similarity measures

The above observation prompted us to consider multidimensional similarity (and dissimilarity) measures. In this situation the filtration value of each simplex is equal to the (appropriately defined) dissimilarity within *all its vertices*. We require that the dissimilarity measure fulfills the filtration property, namely: $dsim(A) \leq dsim(B)$ (alternatively: $sim(A) \geq sim(B)$), for $A \subset B$. This is a crucial property, allowing us to apply persistent homology. Importantly, the multidimensional dissimilarity values are computed from the input data, and cannot be inferred from the pair-wise dissimilarity measure.

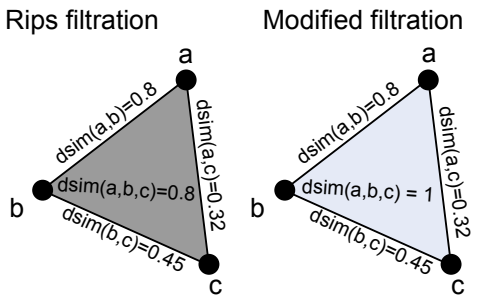

Figure 2: Comparison of the standard Rips filtration (left), where each simplex is assigned the maximum value of its faces, and the multidimensional dissimilarity measure (right). We consider normalized vectors: $a = (0.5, 0, 0.866)$, $b = (0.4, 0.916, 0)$, $c = (0, 0.6, 0.8)$. Once the three edges appear in the Rips filtration, the triangle also appears. On the right, it need not be true – the dissimilarity value of three vertices might be 1 (0 similarity). This is natural: the three objects (a,b,c) might not be similar *as a triple*.

Figure 2 compares the standard Rips filtration with the new definition. Note that the filtration values of higher-dimensional simplices are larger than in the Rips filtration. Therefore, for a chosen dissimilarity threshold, Rips filtrations contains more simplices. Also, at dissimilarity threshold equal to 1, the complex contains all possible simplices. There is no point in explicitly storing such simplices, because we know that every cycle is eventually filled-in.

Additionally, in the case of the Rips filtration, each simplex (of dimension at least 1) has at least one face with the same filtration value. This generates spurious persistence pairs of 0 lifetime, which carry no information. In [19], we employed discrete Morse [9, 12, 11] theory to prune such simplices during the construction of the complex, which is now unnecessary.

Algorithm 1 construct a simplicial complex, filtered by the multidimensional dissimilarity values. For a given threshold, it is a subcomplex of the Rips complex, so the construction is similar. We build the complex dimension by dimension, as in [19]. If the dissimilarity of the vertices of the d-simplex N is below the threshold ϵ, N it is added to constructed complex. Note that all faces of N are present, because they have lower dissimilarity. Alternatively, if computing the dissimilarity is costly, checking if any of the faces of N is missing, allows us to discard N without computing its dissimilarity.

The advantage of proceeding by dimension is once a simplex is discarded, all its potential co-faces are also automatically discarded. In Section 8 we analyze the practical behaviour of our algorithm, which shows that the constructed complex has much smaller size and dimension, compared to the Rips (or clique) complex. For this reason we do not use efficient methods for finding maximal cliques, such as [7]. In the worst case, however, the

presented algorithm has exponential complexity, as the output contains all cliques of the graph.

Algorithm 1 Constructing a simplicial complex filtered by a multidimensional dissimilarity measure.

Input: Point cloud P, measure dsim, ϵ, maximal dimension dim
Output: Filtered complex K

```
(K^0, K^1) := graph over P weighted by dsim
Let nr(v) be the unique index of each vertex v ∈ K^0.
for d = 2... dim do
    K^d = ∅
    for each d-simplex S in K^{d-1} do
        for each vertex e adjacent to any vertex of S do
            if nr(e) > max_{v∈S}{nr(v)} {Note that each simplex is created and added
            only once and that e ∉ S.} then
                N = S ∪ e {N is a new d+1 simplex.}
                if dsim(N) ≤ ε then
                    K^d = K^d ∪ N {All faces of N are already in K. }
return K = (K^0, ..., K^dim)
```

Summarizing this section, the input point cloud can be viewed as a simplicial complex, filtered by an appropraite multidimensional dissimilarity measure. We presented an algorithm, constructing such a complex, where each simplex is assigned a value equal to the dissimilarity among all its vertices.

5 Example, topology in text mining.

In the ongoing research, we build topological tools to robustly analyse and compare text data. The goal is to find meaningful topological patterns. This information can help understand the global structure of the data. In a longer perspective, this knowledge can be used in conjunction with the standard methods, improving the quality of information-retrieval systems.

5.1 Vector space model

We start with describing a way to map textual data into a representation which allows us to use topological tools. Vector space model is a standard tool in information retrieval and data mining [15]. A *corpus*, i.e. collection of text documents, is mapped into a collection of points (or vectors) in $\mathbb{R}^n$. These vectors are the so-called *term-vectors* and each of them represents a single document, as described below. Each dimension of this space corresponds to a single word (or *term*).

Each document in a corpus, is associated with a term-vector [15] containing words characteristic of this document. In practice from 10 to 50 words

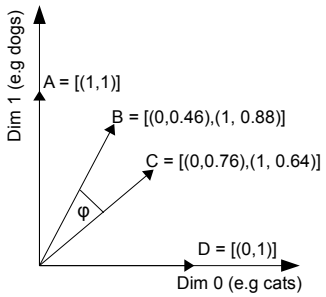

Figure 3: Example of the vector space model. A two-dimensional space is shown, which means that only two different words are extracted from all documents. The similarity between vectors B, C equals $cos(\varphi) = 0.46 \cdot 0.76 + 0.88 \cdot 0.64 = 0.91$.

are extracted, roughly encapsulating the *topic*. Each term t contained in some document d in corpus D is weighted according to the standard tf-idf [15] technique: $w(d,t) = tf(d,t) \cdot idf(t)$, where $tf(d,t)$ is the number of occurrences of word t in document d, and $idf(t) = log\frac{|D|}{|\{p \in D : t \in p\}|}$. Thus, more frequent words in a document are weighted higher but this is offset by the *global* popularity of a given term. By P we denote the array of term-vectors representing all the documents of the corpus D.

Each term in the corpus is assigned a unique index, which represents the term. Term-vector $d \in P$ is compactly stored as a sparse vector: we explicitly represent only the coordinates with non-zero weights. The actual data-structure representing term-vector d is simply an array of pairs (index of t, $w(d,t)$). See Figure 3 for a simple example. Note that, for brevity, we often identify a document with its term-vector.

We use the so-called *cosine similarity measure*, which is a standard text mining tool used to compare documents [15]. The similarity between two documents (represented by term-vectors a, b), is given by $sim(a,b) := cos(\angle(a,b)) = \frac{\langle a,b \rangle}{||a||||b||}$. We work on normalized (according to Euclidean norm) term-vectors and equivalently compute similarity as:

$$sim(a,b) = \langle a,b \rangle$$

Due to very high extrinsic dimensionality, equal to the number of unique words, the space is very sparse, or empty, in practice. Another observation is that the number of keywords extracted from each document is relatively small. So, for each term-vector, the number of nonzero coordinates is small, compared to the number of zero coordinates. Therefore, the similarity between two term-vectors should be zero most of the time, since the support of these vectors is disjoint.

5.2 Multidimensional extension of cosine similarity measure

We propose the following extension of the cosine similarity measure. Assuming $\{V^j\}$ is a collection of L_2-normalized vectors, and V_i^j is the $i-$th

coordinate of $j-$th vector, we define the extended measure as:

$$Sim(V^1, ..., V^k) = \sum_i (\prod_{j=1}^{k} V_i^j)$$

Note that for $k = 2$, this definition agrees with the standard cosine measure. Unlike the standard definition, there is no obvious geometric interpretation. There is a compelling reason for using this definition, rather than taking the using the standard Rips filtration: The new definition correctly models the situation (and its generalizations) when a triple of objects is pairwise similar, but is not similar as a triple, as depicted in Figure 2.

We prove that Sim satisfies the filtration property, that is: whenever $A \subseteq B$, $Sim(A) \geq Sim(B)$. Denote $A = A^1, ..., A^k$ and $B = A^1, ..., A^k, B^{k+1}, ..., B^h$. For each i, we put $P_i(A) := \prod_{j=0}^{k}(A_i^j)$ and $P_i(B) := \prod_{j=0}^{h}(B_i^j) = P_i(A) \prod_{j=k+1}^{h}(B_i^j)$, with $\prod_{j=k+1}^{h}(B_i^j) \leq 1$ because of the normalization. Therefore $P_i(A) \geq P_i(B)$, and $Sim(A) \geq Sim(B)$.

6 Example, feature space of images.

While the main motivation of our work was the presented application in text mining, the described techniques can be applied more widely. In general, the presented techniques can be used to analyze topology of arbitrary feature-spaces, provided an appropriate multidimensional similarity measure can be defined. The described *vector space model* relies on the concept of the feature-space. In a sense our approach is similar to [16], but we focus more on the topological features rather then the geometry of the feature-space.

In this section we give another example, more suited for image understanding. Let us consider a set of images, assuming each image is characterized by a vector of binary features. These features could be label, i.e. words characterizing their contents. Alternatively, the features could be obtained automatically, by means of automatic feature detectors.

In case of weighted feature-vectors, the described cosine similarity measure can be used. In the case of binary vectors, we can use the well-known *Jaccard index* [10], when binary feature-vectors correspond to sets. For two sets of features, A, B, it is defined as follows:

$$j(A, B) := \frac{|A \cap B|}{|A \cup B|}.$$

6.1 Multidimensional Jaccard index

For P, being a collection of feature-sets (binary feature-vectors), we introduce the multidimensional Jaccard index, describing the similarity within a collection of objects $X \subset P$. It is defined as:

$$J(X) := \frac{|\bigcap X|}{|\bigcup X|}$$

It is easy to prove that this measure fulfills the filtration property. Let us consider a nonempty $A \subseteq B$. Clearly $|\bigcap(B)| \leq |\bigcap(A)|$ and $|\bigcup(B)| \geq |\bigcup(A)|$, therefore $J(A) \geq J(B)$, as required.

7 Interpreting the topology

In this section we hypothesize about different usages of the presented techniques. The computed topological information can be used directly or indirectly. The main idea is related to the methodology of manifold learning, but we do not aim at reconstructing the lower dimensional structure. Instead, we only extract the most prominent topological features, namely the features of high persistence.

In the direct approach we can attempt to interpret the computed topological descriptors. First note that the 0-dimensional simplices represent the objects, and higher dimensional simplices represent the similarity between subsets of objects.

Consider a p-cycle, which is composed of p-simplices. When this cycle is created, it means that a subset of objects is inter-related with respect to the p-dimensional dissimilarity measure, at a certain dissimilarity threshold. As we increase the threshold, less and less similar objects are considered related. The cycle is eventually killed at a higher threshold, by being filled-in by some (p+1)-dimensional simplices. These simplices represent (d+1)-dimensional similarity criterion. Intuitively, the p-dimensional similarity information carried by the p-cycle is made trivial by a stronger similarity criterion. Persistence, which is the lifetime of the cycle, quantifies the relatedness of the objects represented by the vertices of the cycle.

This sort of analysis is different than, for example, k-means clustering, which captures local structure, using only pairwise relationships between objects. In particular complicated homology, can be indicate that simple clustering methods might fail. For example consider Figure 1, where the loops would be cut into several clusters, which might not be the correct interpretation.

At the current stage, direct interpretation of high-dimensional topology is not our aim. Instead, we can use this information indirectly. As mentioned in Section 3.2, the procedure is stable, therefore the distance between persistence diagrams actually quantifies the differences in topology of the data.

One usage is to treat the persistence diagram as a descriptor of the data. Specifically, one can compare different datasets (or different parts of data) by comparing their persistence diagrams. Similarly, one can construct a

model of a certain type of data (for example: images of cancer tissue), and test other datasets against it, by comparing their persistence diagrams.

Additionally, topological analysis can point to parts of the feature-space which are not populated. This might be related to an inherent feature of the dataset or missing data due to under-sampling. In both cases this might be an interesting information in case of interactive data exploration.

8 Experiments and efficiency

We have developed a C++ implementation which includes the methodology outlined above. In the final step of computations, we the PHAT library [1] for persistent homology. It implements various methods, including the the *twist algorithm* [3], also used for the original experiments [19]. We sample the corpus of the English Wikipedia [20], processed using the gensim library [21].

There are two main parameters of our software. Parameter *dim* controls the maximum dimension of the constructed complex, allowing for computing persistent homology up to dimension $dim - 1$. The second parameter is $\epsilon \in [0, 1]$, which means that only simplices with sufficiently low dissimilarity are included.

8.1 Analyzing Rips complex filtered with one-dimensional similarity

Using the efficient algorithms from the PHAT library we rerun the experiments from [19], using the Rips filtration based on the standard, pair-wise cosine measure. Standard methods again exhibited roughly quadratic complexity (in the size of the complex) for dimensions ≥ 3 (see Figure 4:left). We use PHAT v1.2.1 command line tool with the default *twist* algorithm, the default strategy for column operations, and the *dualize* option. The last option is crucial, as it effectively computes persistent co-homology, which is much more efficient for filtered Rips complexes. This does not affect the results [17], but yields execution faster by several orders of magnitude. The comparison was limited by the inefficiency of the previous method. For larger complexes, of roughly 0.8 and 1.44 million simplices, the respective running times were only 1s and 2.2s.

8.2 Analyzing a complex filtered with multidimensional similarity

Using the multidimensional similarity measure yields smaller complexes, as shown in Figure 4:right. This is a consequence of the property described in Figure 2. Higher-dimensional simplices with dissimilarity threshold ≤ 0.8 seem to be quite rare. This is not surprising: the larger subsets we consider,

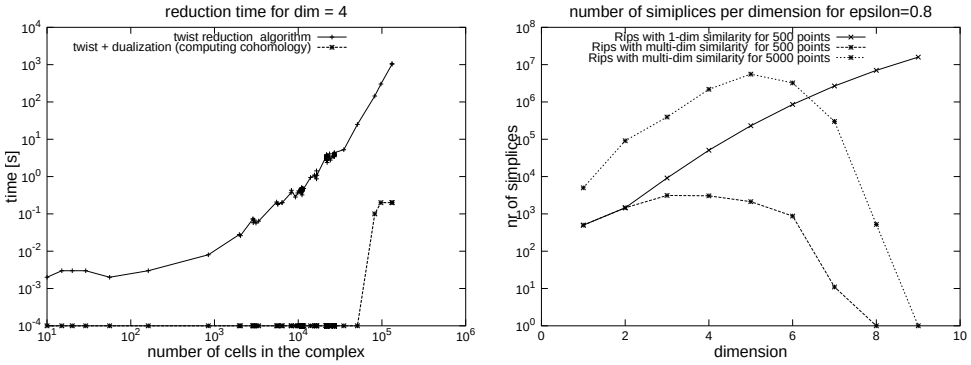

Figure 4: **Left:** Runtime (in seconds) on a log-log scale for two algorithms: the twist algorithm and the most suitable algorithm from the PHAT library. The new approach is orders of magnitude faster, with only 0.2 seconds (instead of 1000 seconds) for the largest dataset. **Right**: Number of simplices in each dimension of a generated complex. For the smaller dataset, the new construction gives significantly smaller complexes (in dimension ≥ 2). In particular, we see that the sequence is unimodal, and stabilizes at zero. On the other hand, the size of the Rips complex increases monotonically in this range of dimensions, resulting in a very large representation. For a larger dataset we see a similar behaviour (constructing the Rips complex for this range of dimensions was infeasible in this case).

the harder it is for them all to be similar. Because there are no simplices in dimension ≥ 10, we know that homology is trivial in these dimensions.

Summarizing this section, we note that there two aspects which affect the performance. Using the new algorithms for persistent homology removes the main computational bottleneck. Additionally, using the introduced multidimensional dissimilarity measure yields much smaller complexes in practice.

9 Summary and further thoughts

We think it would be useful to devise different multidimensional similarity measures, tailored for specific application. Used in conjunction with existing topological methods, such measures not only give a more natural interpretation of the data, but provide a more compact representation as well.

An important conclusion is the fact that current generation of persistence algorithm can handle much larger data, allowing for scalable solutions. Efficiency used to be a significant bottleneck of the previous implementation. Overcoming this limitation gives hope for applications in real-world situations.

Acknowledgments

Both authors are supported by Google Research Awards programme. We thank Prof. Marian Mrozek and Witold Jarnicki for their supervision, and Herbert Edelsbrunner, Leonidas Guibas, Ulrich Bauer and Jan Reininghaus for helpful comments and discussions. H.W. is also supported by a Foundation for Polish Science IPP Programme "Geometry and Topology in Physical Models". P.D is supported by federal contracts FA9550-12-1-0416 and FA9550-09-1-0643.

References

[1] U. Bauer, M. Kerber, J. Reininghaus, *Clear and Compress: Computing Persistent Homology in Chunks*, TopoInVis 2013.

[2] G. Carlson, *Topology and Data*, Bulletin of the AMS, 46(2), pp. 255-308 (2009).

[3] C. Chen, M. Kerber, *Persistent homology computation with a twist*, 27th European Workshop on Computational Geometry (EuroCG 2011) (2011).

[4] F. Chazal, V. de Silva, S. Oudot, *Persistence stability for geometric complexes*, arXiv:1207.3885v1, 2012.

[5] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, *Stability of Persistence Diagrams*, Discrete Comput. Geom., 37(1):103-120, 2007.

[6] H. Edelsbrunner, J. Harer, *Computational Topology. An Introduction.* Amer. Math. Soc., Providence, Rhode Island (2010).

[7] D. Eppstein, D. Strash, *Listing all maximal cliques in large sparse real-world graphs*, 10th International Symposium on Experimental Algorithms, arXiv:1103.0318, (2011)

[8] A-X. Feng, C-H. Fu, X-L. Xu, A-F. Liu, H. Chang, D-R. He, G-L. Feng, *An Empirical Investigation on Important Subgraphs in Cooperation-Competition networks*, Science (2011).

[9] R. Forman, *A User's Guide To Discrete Morse Theory*, Sminaire Lotharingien de Combinatoire, Vol. B48c, pp. 1-35 (2002).

[10] P. Jaccard, *The distribution of the flora in the alpine zone*, New Phytologist 11: 3750., 1912.

[11] T. Lewiner, *Geometric discrete Morse complexes*, PhD Thesis (2005)

[12] T. Lewiner, H. Lopes, G. Tavares, *Toward Optimality in Discrete Morse Theory*, Experiment. Math. Volume 12, Number 3 (2003), pp 271-286.

[13] X. Polanco and E.S. Juan, *Text Data Network Analysis Using Graph Approach*, Proc. of InSciT, pp. 586-592 (2006).

[14] V. Robins, P. J. Wood, A. P. Sheppard, *Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images*, IEEE Trans. Pattern Anal. Mach. Intell. 33, pp. 1646-1658 (2011).

[15] G. Salton, A. Wong, C.S. Yang, *A vector space model for automatic indexing*, Commun. ACM 18(11), pp. 613–620 (1975).

[16] B. Scholkopf et al., *Input space versus feature space in kernel-based methods*, IEEE Trans on Neural Networks, 10(5), 1999.

[17] V. de Silva, D. Morozov, M. Vejdemo-Johansson, *Dualities in persistent (co)homology*, Inverse Problems 27 124003, doi:10.1088/0266-5611/27/12/124003, 2011.

[18] A. Zomorodian, *Fast construction of the Vietoris-Rips complex*, Computers & Graphics, Vol. 34, Nr. 3, pp. 263-271 (2010).

[19] H. Wagner, P. Dłotko, M. Mrozek, *A Computatinal Topology in Text Mining*, CTIC 2012, LNCS Volume 7309, 2012, pp 68-78

[20] *English Wikipedia corpus* `http://dumps.wikimedia.org/enwiki/`.

[21] *Gensim Library* `http://radimrehurek.com/gensim/`.