Appendix A
Algorithms in Topological Data Analysis

A.1 Computing Persistent Homology

To explicitly compute the homology of a simplicial complex, one needs to choose bases for $C_k(X)$ and $C_{k-1}(X)$ and then find the image and kernel of the boundary map

$$\partial_k \colon C_k(X) \to C_{k-1}(X).$$

This can be done via representing the boundary map as a matrix with respect to the chosen bases and putting this matrix in Smith normal form. In order to compute persistent homology, one has to choose compatible bases for each chain group simultaneously; this is what is behind the standard algorithms for computing the persistent homology (recall Section 2.7 or see for example [551]).

These algorithms are $O(n^3)$ in the number of simplices in the complex, and while they are often linear in practice, it is not hard to construct explicit filtrations that achieve the cubic bound [361]. As discussed in Section 2.7, if the feature scale is close to the diameter of the data, the Vietoris-Rips filtration will have exponentially many simplices. There have been some efforts to achieve better performance by simplifying the complex; [455] describes how to build a hierarchical collection of approximations to suitable finite metric spaces such that for any given accuracy the computation time is linear in the number of points X, and [549] uses simplicial collapses to reduce the complex without changing its homotopy type.

However, the best performance to date has been achieved by work that uses a series of optimizations of the basic algorithm, most notably the use of persistent cohomology in Ripser [39], which is currently the fastest and most memory-efficient available implementation (for Vietoris-Rips complexes).

A.2 Software for Persistent Homology and Mapper

We begin by describing the software available for persistent homology. There are essentially two categories of efficiency: the first group does not work for large complexes but in general has more functionality, and the second group provides the state of the art performance. A very nice summary of the situation (with detailed performance comparisons) is given in [413]. All of the packages we discuss are under active development at the time of writing.

1. **Javaplex:** handles zigzag persistence and construction of the witness complex [490].
2. **Dionysus:** handles vineyards, zigzag persistence, and persistent cohomology [360].
3. **Perseus:** handles cubical complexes as input [376].

For large data sets, there has been a recent revolution; most notably, Ripser and Eirene work on remarkably large complexes (billions of simplices).

1. **Gudhi:** handles the witness complex, subsampling, persistent cohomology, cubical complexes [341].
2. **Dipha:** distributes computation across many parallel nodes [40, 41].
3. **Ripser:** fastest performance available, only compatible with Vietoris-Rips so far [39].
4. **Eirene:** very good performance (comparable to Ripser), supports identification and rendering of specific cycles that represent homology [238, 239].

Software for multidimensional persistence has been considerably more limited, until the recent development of the excellent Rivet tool.

1. **Rivet:** supports novel simplification techniques and rendering options [324, 325].

A number of the statistical techniques for topological data analysis, most notably persistence landscapes, are now available in free software packages.

1. **Hera:** supports fast computation of bottleneck and Wasserstein distances [289, 290].
2. **TDA statistics R package:** supports confidence sets, bootstrapping, persistence landscapes, and distance to a measure. Incorporates Gudhi, Dionysus, and PHAT [171]. (PHAT is a library for persistence computation [43, 44].)
3. **Persistence Landscape Toolbox:** supports various statistical operations in the context of persistence landscapes [77, 78].

The situation for Mapper is somewhat more limited amongst free software; far and away the best implementation is the Ayasdi version, which requires a license and contacting the company in order to download the software.

1. **Python Mapper:** an experimental package that provides both a GUI interface and a python package [365].
2. **TDA Mapper:** an R package exposing Mapper functionality [402].
3. **KeplerMapper:** an experimental package that provides a python library [447].
4. **Ayasdi:** commercial software for Mapper, supporting a wide range of filters, a well-developed GUI, and elaborate output renderings [267].