Incremental Matrix Reduction

Anindya Moitra

1 The Background

This document addresses one of the primary challenges of computing persistent homology on streaming data by a fully incremental approach. As a data stream is a potentially infinite sequence of data objects, the entire stream cannot be stored in the memory typically available to a computer. Therefore, the computation of persistent homology on streaming data requires an incremental approach. A couple of computational models for applying persistent homology on data streams are being developed as two separate projects that involve partially incremental approaches. In particular, consistent with the standard computational paradigm [1] for processing data streams, each of those two models consists of two principal components: (i) *online*, and (ii) *offline*. However, yet another (*i.e.*, a third) computational model is developed by a *fully online* or *fully incremental* approach in this project.

A key requirement for developing such a fully incremental model for persistent homology is the ability to perform the *Gaussian elimination* (also called the *reduction*) of the boundary matrix [2, 3] by an incremental algorithm. The Gaussian elimination step is performed during the offline component (*i.e.*, as a *batch processing* mechanism) in the previous two models for computing persistent homology on streaming data. This document develops the theoretical foundation for performing the *Gaussian elimination* by an incremental algorithm.

It is worth mentioning that while computing persistent homology on data streams is one of the primary target applications of the incremental Gaussian elimination algorithm, it will have other important applications as well. For example, due to the large size of the *complex* constructed on a point cloud, the dimension of the boundary matrix increases exponentially with the number of data objects on which persistent homology is being computed. As a result, reducing the boundary matrix by a batch processing algorithm becomes prohibitive even for 'static' (*i.e.*, non-streaming) data sets of moderate size (such as, data sets with up to a few thousands of objects, depending on the memory available to the computer). The incremental algorithm developed by this project will help in the Gaussian elimination of the boundary matrix where this is not possible by batch processing mechanisms due to the size of the matrix.

2 The Problem

The standard algorithm [2,3], as described in [4–7] among others, computes the persistence of a filtration [8] by *reducing* its boundary matrix ∂ to a column-echelon form R. Usually, the entire boundary matrix or a simplified data structure thereof is processed in the memory while the standard algorithm is executed. This approach is not desirable when computing persistent homology on streaming data by a fully incremental mechanism. When working with data streams processed by a fully incremental model, one would want to add a simplex σ to the already reduced matrix R without having to recompute the reduction of other columns due to the addition of σ. Ideally, only the column of σ should be reduced as the simplex σ is added to the filtration.

Kerber *et al.* [7] introduced a streaming algorithm for reducing the boundary matrix based on optimized versions of the standard algorithm [2, 3]. However, their algorithm assumes that the entire data set or filtration is available on disk. Therefore, the total ordering of the simplices is predetermined. The next simplex added to R has a higher weight than any of the previously added simplices.

In a real streaming application, the entire point cloud or filtration is not available at any time. Hence, every time a new simplex σ is added to the filtration, the indices of those simplices that have weights higher than that of σ are incremented by one. In other words, in a real streaming environment, new simplices do not arrive from a sequence of simplices sorted according to their increasing weights. Therefore, the algorithm of [7] can not be applied to real-world streaming applications.

In this document, we prove that a streaming or incremental version of the standard algorithm exists. In particular, we prove that even when a simplex σ that changes the indices of other simplices is added to an already reduced matrix R, it is possible to only reduce the column of σ without recomputing the reduction of other columns.

3 A Solution

3.1 Incremental Matrix Reduction Algorithm

An incremental variant of the standard algorithm can be described as follows. Let us assume that a boundary matrix ∂ is already reduced to the column-echelon form R. When a new simplex σ arrives, we compute its index j with respect to the existing filtration. We also compute the row r and the *unreduced* column c of σ. c specifies the facets[1] of σ, while r contains only zeroes.

Theorem 3.1. *The algorithm described above computes the correct persistence intervals.*

Proof. Assume that the new simplex σ is added to an existing filtered simplicial complex K. An existing simplex $\tau \in K$ can become a facet of σ. However, τ can never be a cofacet[2] of σ. In other words, an existing simplex of K can be a facet of a new simplex, but the new simplex can not be a facet of an existing simplex. This is due to the definition of the simplicial complex K: every face of an existing simplex τ must already be contained in K.

The j-th column of the boundary matrix ∂ encodes the facets of the simplex σ_j, and the i-th row of ∂ encodes the cofacets of σ_i. Since the new simplex σ can not have an existing cofacet, r contains only zeroes.

The standard algorithm iterates over the columns of the boundary matrix ∂ from left to right, reducing the columns by *left-to-right column additions*. Therefore, the newly added j-th column c needs to be reduced according to the standard column reduction procedure described above.

We call R reduced when all the pivots are in unique rows. It is well-known that, although ∂ does not have a unique reduction, the pivots of all its reductions are the same. When a new data point arrives from the stream, the corresponding new simplices are created by iteratively adding the cofaces upto the maximum dimension k. The data point itself creates a 0-simplex. When that 0-simplex is added to R, it has no effect on the existing persistence intervals. This is because the column of a 0-simplex contains only zeroes. The addition of the subsequent cofaces of the 0-simplex can not introduce a pivot that equals the pivot of a higher column in R, because:

- with the addition of a new simplex at column j, the existing pivots greater than j are incremented by one (due to the accompanied addition of a row at the position j),

[1] A facet is a co-dimension one face of a simplex.

[2] A cofacet is a co-dimension one coface of a simplex.

- R is upper-triangular,
- the data points (*i.e.*, the 0-simplices) from the stream are assumed to arrive sequentially, and thus, have a natural total ordering,
- the simplices always have a total ordering dictated by the filtration.

Therefore, when a column c that corresponds to a new simplex is added to R at the position j, the new column can not have the same pivot as a column greater than j. The new column j and those to its right will have unique pivots, which implies that we would not need to reduce the columns to the right of j.

□

4 Discussion

The incremental matrix reduction algorithm describer here could be optimized in several ways, such as those given in [7]. An efficient implementation of the algorithm would require data structures such as sparse matrix [4], T-array [3] or dictionary [7].

References

[1] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. P. L. F. de Carvalho, and J. ao Gama, "Data stream clustering: A survey," *ACM Computing Surveys*, vol. 46, no. 1, pp. 3.1–13.31, Oct. 2013.

[2] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, ser. FOCS '00. Washington, DC, USA: IEEE Computer Society, 2000.

[3] A. Zomorodian and G. Carlsson, "Computing persistent homology," *Discrete & Computational Geometry*, vol. 33, no. 2, pp. 249–274, Feb. 2005.

[4] H. Edelsbrunner and J. Harer, *Computational Topology, An Introduction*. American Mathematical Society, 2010.

[5] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 1, Aug. 2017.

[6] C. Chen and M. Kerber, "Persistent homology computation with a twist," in *Proceedings 27th European Workshop on Computational Geometry (EuroCG'11)*, 2011, pp. 197–200.

[7] M. Kerber and H. Schreiber, "Barcodes of towers and a streaming algorithm for persistent homology," *Discrete & Computational Geometry*, Oct. 2018. [Online]. Available: https://doi.org/10.1007/s00454-018-0030-0

[8] A. Zomorodian, "Fast construction of the vietoris–rips complex," *Computer and Graphics*, pp. 263–271, 2010.