AMAT 584 Lec 39, 5/1/20

Today: Remarks on the Scalability of Persistent Homology computation

Stability of persistent homology (if time permits)

Proposition: If $F$ is a filtration such that $F_{max}$ has $n$ simplices, then our algorithm computes the barcodes of $F$ in $O(n^3)$ elementary operations, and $O(n^2)$ units of memory.

Note: $O(f(n))$ means "at most $cf(n)$, for $c$ some constant independent of $n$, for all $n$ sufficiently large."

While this theoretical result is a good starting point for understanding the scalability of persistent homology computation, it does not come close to capturing the reality of how these computations scale.

First, it should be emphasized that the $O(n^3)$ bound is a worst-case bound, and one usually sees much better (linear) scaling in practice.

## What size data sets can we handle?

There's no simple answer.

The answer depends on many things, e.g.

- Which barcode(s) am I trying to compute? $0^{th}$? $1^{st}$? $2^{nd}$?
- Which filtration am I considering?

Recall: To compute barcodes up to homology dimension $k$ I only need the simplices in $F_{max}$ of dimension at most $k+1$.

### Vietoris-Rips

For $F$ the VR filtration of a point cloud with $m$ points, $F_{max}$ has $n = 2^{m-1}$ simplices. ⇒ unless $m$ is really small, we cannot compute all the barcodes of $F$.

If we just compute the first $k$ barcodes, the number of simplices $n$ required by or algorithm

$$m + \binom{m}{2} + \binom{m}{3} + \cdots \binom{m}{k+2} < c m^{k+2}$$ for some small constant $c$.

This is much smaller than $2^{m-1}$, though it still increases quickly as $k$ increases.

Thus, we typically take $k=2$ or $k=1$.

There are several slick tricks one can do to reduce the time and memory cost of VR barcode computations.

The state-of-the-art is Uli Bauer's Ripser software.
(http://ripser.org)

For Rips filtrations, "naive" computations of the 1st persistence barcode, using state-of-the art software became difficult for, say, 5000-10,000 points on a recent consumer-grade laptop.

Memory, not, speed is almost always the issue, so using a computer with lots of memory can raise the ceiling a little bit.

## Alpha/Delaunay Filtrations

For $F$ the Delaunay filtration of $m$ points in $\mathbb{R}^d$, $F_{max}$ has $n = O(m^{\lceil d/2 \rceil})$ simplices. ($\lceil \ \rceil$ denotes the ceiling function, e.g. $\lceil 2 \rceil = \lceil 3/2 \rceil = 2$.

A typical case is $d=3$. Then $n = O(m^2)$.

This is a pretty small bound; for data in $\mathbb{R}^3$, we can handle hundreds of thousands of points.

## Truncation

To keep the number of simplices small, we can also truncate the filtration only up to some fixed index.

If I know something in advance about the structure of my data, I may be justified in truncating. But often this is undesirable.

With some truncation, we can handle much larger data sets with the Rips construction.

## Approximate Computation

There are both simple and sophisticated approximation methods that can significantly improve scalability

For Rips complexes, a recent software Sparips is recommended.

Quality of the approximation depends on the intrinsic dimensionality of the data (doubling dimension).

Computational cost is a serious limitation of TDA methods, at least for high-dimensional data!

But for many types of data, the methods scale well enough to be very useful.

There are a lot of tricks that can be leveraged, and state-of-the-art methods keep getting better.