Finding the minimal number of points sampled from $\mathbb{S}^n$ get a persistent interval in H_n of lifetime k

Rolando Kindelan Nuñez

November 2024

1 Problem

Given an n-dimensional sphere unit $\mathbb{S}^n$, the homology groups are $\mathbb{Z}$ in dimension 0 and n, and 0 otherwise. Now, suppose we sample points from uniform distribution in $\mathbb{S}^n$. If we are able to sample many many many points, we expect then a persistence interval $[0, 1]$ in dimension n.

The question is, how many points ($nPts$) do we need to sample in order to have, with a high probability, an interval of a length $k \in [0, 1]$ in n dimensional persistence. We can, at the beginning, set k to 0.5,

Now, the tricky part is what do we mean by high probability? Let us define it in Monte Carlo fashion - say that, in the collection of N samples, 95% of cases we observe such a interval. So, is $N = 100$, if I sample n points 100 times, 95 times we will observe a persistence interval of a length greater of equal k in dimension n.

Find n as a function of k and the significance level (95%).

2 Proposed Solution

Our proposed solution considers the following variables:

1. Maximal hyper-sphere dimension: $Dmax = 30$
2. Monte-Carlo samples: $N = 100$
3. % Confidence: $C = (N \cdot 95)//100$
4. Expected failures: $ef = N - C$
5. Desired persistence: $k = 0.5$
6. Filtration type: $ftype \in \{\alpha - complex, Sparse\ Rips\}$

Our Algorithm 1 basically iterates per dimension d. For each dimension we traverse a collection of candidate points $nPts \in [2, 100000]$. For each of $nPts$ values we perform the Monte-Carlo simulation sampling $nPts$ points from $\mathbb{S}^d$, compute the homology groups on a $ftype$ filtration and see if we have an interval $(birth, death) \in H_d$ such that $persistence(birth, death) \geq k$.

There are several observations to pointed out:

- Since we need a given significance level C, the candidate $nPts$ only has allowed to fail f times trying to get a desired persistence interval. Therefore, by tracking the failures we reject $nPts$ values that will not satisfy the required significant level until we find the first one that matches the requirements.
- As a d-cycle can be decomposed into multiple (d-1)-chains, it is required a large number to create a d-cycle than the number required to have a (d-1)-chain. This implies, that it is safer to start searching for a d-cycle with the minimal number of points encountered on dimension d-1 that we will call old_npts. It is also safer to use the same amount as increment when a given number of points is rejected, hence we move on the point buckle with $nPts+ = old\_npts$. It may be the case that $\exists np \in [nPts, nPts + old\_npts]$ for which we can find a desired persistence interval, but we does not care these cases. We could do a binary search to find the actual minimal number of points.

Considering Algorithms 1 with default values and constructing α-complex based filtration we got the results showed in Table 2.

Table 1: Results obtained with $n \in [1, 4]$ and considering $nPts$ increment as the older number of points of previous dimension. By the higher confidence obtained it seems that there are a lower number of points that can reach the desired confidence of 95%.

n	$nPts$	$confidence$	$complex$
1	34	95	α-complex
2	170	98	
3	510	96	
4	1530	97	

Algorithm 1 find_points(ftype, Dmax, N, Sl, k).

Require: Let $ftype$ be the desired filtration type, $Dmax$ be the maximal hypersphere dimension, N be the monte-carlo number of trials, Sl be the significance level, and k be the desired lifetime in the desired persistence interval.

Ensure: A collection of 3-tuples $R = \{(n, nPts_n, C_n)\}_{n\in[1,Dmax]}$ relating the dimension, minimum number of points and confidence level.

```
1: $C \leftarrow (N \cdot Sl)//100$
2: $ef \leftarrow N - C$
3: $R \leftarrow \{\}$
4: $n \leftarrow 1$
5: $old\_npts \leftarrow 2$
6: for $n < Dmax$ do
7:     $nPts \leftarrow old\_npts$
8:     for $nPts < Dmax$ do
9:         $f \leftarrow 0$
10:        for $t < N$ do
11:            $ans \leftarrow analyze\_persistence(nPts, n, k, ftype)$
12:            if $ans = False$ then
13:                $f \leftarrow f + 1$
14:            end if
15:            if $f > ef$ then
16:                $Break$                                   ▷ increase the $nPts$
17:            end if
18:            $t \leftarrow t + 1$
19:        end for
20:        if $f > ef$ then
21:            $nPts \leftarrow nPts + old\_npts$
22:        else
23:            $R \leftarrow R \cup \{(n, nPts, N - f)\}$
24:            $old\_npts \leftarrow nPts$
25:            $Break$                                       ▷ increase the $n$
26:        end if
27:    end for
28:    $n \leftarrow n + 1$
29: end for
```

Algorithm 2 analyze_persistence(nPts, n, k, ftype).

Require: A number of points $nPts$, the desired dimension n of the $\mathbb{S}^n$.

Ensure: Says $True$ if we found a persistence interval in H_n with lifetime higher than k, and $False$ otherwise.

```
1: $P \leftarrow sampling\_nsphere(nPts, n)$
2: $Diags \leftarrow compute\_ftype\_persistence(P, n)$
3: for $(d, (birth, death)) \in Diags$ do
4:     if $d = n$ and $persistence(birth, death) \geq k$ then
5:         return $True$
6:     end if
7: end for
8: return $False$
```

Algorithm 3 sampling_nsphere(nPts, n).

```
Require: A number of points $nPts$, the desired dimension of the nsphere $n$.
Ensure: A collection $P \in \mathbb{R}^{n+1}$ of points in the boundary of $\mathbb{S}^n$.
1: $F \leftarrow np.random.randn(nPts, n+1)$
2: $P \leftarrow F/np.linalg.norm(F, axis = 1, keepdims = True)$
3: return $P$
```

3 Parallel Approach

In this section, we take advantage of multicore architectures on Algorithm 1. Let us say that a given computational machine has p processors. We can execute p from the N trials in parallel, if $p > ef$ then we can discard a candidate number of points faster in $O\left(\left\lceil \frac{ef}{p} \right\rceil \cdot C\right)$, with C the complexity of *analyze_persistence*$(\cdots)$, which involves constructing the filtration and computing persistent homology. In general, we can accept a number of points in $O\left(\left\lceil \frac{N}{p} \right\rceil \cdot C\right)$.

We have implemented this approach using a pool of processes through the python multiprocessing package. Resulting in the following table:

Table 2: Results obtained with $n \in [1, 4]$ and considering $nPts$ increment as the older number of points of previous dimension. By the higher confidence obtained it seems that there are a lower number of points that can reach the desired confidence of 95%.

n	$nPts$	$confidence$	$complex$
1	34	100	α-complex
2	170	100	
3	680	100	
4	2040	100	

It should be reviewed why it provides 100 confidence. I stop the computation of the H_5 for 5-sphere after many hours.

4 Future Work

There are several avenues for future work, few of them captured in this not exhaustive list:

- Enhance the sampling algorithm on the n-sphere (Algorithm 3) to evenly distribute the points. An example could be utilizing Fibonacci lattices or Fibonacci spheres in n dimensions by considering generalized polar coordinates (https://math.stackexchange.com/questions/3291489/can-the-fibonacci-lattice-be-extended-to-dimensions-higher-than-3). This is also known as Fibonacci sampling.
- Unfortunately, the proposed approach actually does not answer the minimal number of points required to create the n-cycle with persistence k, but how dense will our n-sphere should be to produce such a given n-cycle with desired persistence. A direct answer may be obtained considering the problem of creating a n-cycle on the n-sphere, then reporting how many points it takes. An approach we thought of was designing a canonical n-cycle and projecting it to the n-sphere. Then, modify the projected p-cycle by adding or removing p-simplices until we get the desired persistence and report the final number of points. We could not devise any practical approach to address this. Then we read that this problem of generating optimal p-cycles is NP-hard (https://www.cs.purdue.edu/homes/tamaldey/course/CTDA/topic8.pdf).