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

Rolando Kindelan Nuez

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 Naive Approach

Our first and naive 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 point numbers $nPts \in [2, N]$ with $N = 10^{d+1} - old_pts + 1$. 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 iterate the points increasing with $nPts+ = old\_npts$, which can be thought as a linear search on the number of points, as summarized by Algorithm 2. Jumping old_npts points per iteration, may lead to cases where $\exists np \in [nPts, nPts + old\_npts]$ for which we can find a desired persistence interval. We could do a galloping (or exponential) search to find the actual minimal number of points in dimension d starting from old_npts, described as Algorithm 3. This approach may not be always better than the linear one, because despite the number of different point number checking is reduced, each np grows exponentially which effectively increase the time and space complexity of constructing the complex and computing persistence homology. We will consider $find\_points$ as an abstraction of these two approaches, in lower dimensions we recommend linear and in higher dimensions galloping.

Algorithm 1 naive_approach(Dmax, N, C, k).

Require: Let $ftype$ be the desired filtration type, $Dmax$ be the maximal hypersphere dimension, N be the monte-carlo number of trials, C be the confidence level $[0, 100]$, and k be the desired lifetime in the desired persistence interval.

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

$ef \leftarrow N - C$
$R \leftarrow \{\}$
$n \leftarrow 1$
$old\_npts \leftarrow 2$
for $n < Dmax$ **do**
    $nPts \leftarrow old\_npts$
    $old\_npts \leftarrow find\_points(n, old\_npts, N, k, ef, \textbf{OUT } R)$
end for
R

Algorithm 2 find_points_linear(n, old_npts, N, k, ef, OUT R).

Require: Let n be the hyper-sphere dimension, old_npts be the points obtained in previous dimension, N be the monte-carlo number of trials, ef be the expected failures, and k be the desired lifetime in the desired persistence interval.

Ensure: Append a 3-tuple $\{(n, nPts, rate)\}_{n\in[1,Dmax]}$ into R.

$empirical\_upper\_bound \leftarrow 10^{n+1} + old\_npts - 1$ ▷ Experimentally, this bound was never reached
$nPts \leftarrow 0$
$success \leftarrow FALSE$
$fails \leftarrow 0$
while $nPts < empirical\_upper\_bound$ and $answer \neq TRUE$ **do**
    $nPts \leftarrow nPts + old\_npts$
    $answer, fails \leftarrow process\_points(nPts, n, N, k, ef)$
end while
if $answer = TRUE$ **then**
    $R \leftarrow R \cup (n, nPts, N - fails)$
end if
return $nPts$

Algorithm 3 find_points_galloping(n, old_npts, N, k, ef, OUT R).

Require: same arguments of Algorithm 2.

Ensure: Also same output.

$lower\_bound \leftarrow old\_npts$
$upper\_bound \leftarrow old\_npts$
$success \leftarrow FALSE$
$fails \leftarrow 0$
while $upper\_bound < empirical\_upper\_bound$ and $answer \neq TRUE$ **do**
    $upper\_bound \leftarrow upper\_bound * 2$
    $answer, fails \leftarrow process\_points(upper\_bound, n, N, k, ef)$
end while
$min\_points \leftarrow upper\_bound$
$rate \leftarrow N - fails$
while $lower\_bound \leq upper\_bound$ **do**
    $mid \leftarrow lower\_bound + (upper\_bound - lower\_bound)/2$
    $answer, fails \leftarrow process\_points(upper\_bound, n, N, k, ef)$
    if $answer = TRUE$ **then**
        $min\_points \leftarrow mid$
        $rate \leftarrow N - fails$
        $upper\_bound \leftarrow mid - 1$
    else
        $lower\_bound \leftarrow mid + 1$
    end if
end while
$R \leftarrow R \cup (n, min\_points, rate)$
return min_points

Algorithm 4 process_points(nPts, n, N, k, ef).

Require: A number of points $nPts$, nsphere dimension n, trials count N, desired persistence k, and ef expected failures.

Ensure: A 2-tuple $(success, failures)$.

$failures \leftarrow 0$
$t \leftarrow 0$
while $failures \leq ef$ and $t < N$ **do**
    $answer \leftarrow analyze\_persistence(nPts, n, k)$
    if $answer \neq TRUE$ **then**
        $failures \leftarrow failures + 1$
    end if
    $t \leftarrow t + 1$
end while
$success \leftarrow failures \leq ef$
return $(success, failures)$

Algorithm 5 analyze_persistence(nPts, n, k).

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.

$P \leftarrow sampling\_nsphere(nPts, n)$
$Diags \leftarrow compute\_ftype\_persistence(P, n)$
for $(d, (b, d)) \in Diags$ **do**
    if $d = n$ and $(\sqrt{d} - \sqrt{b}) \geq k$ **then**
        return $TRUE$
    end if
end for
return $FALSE$

Algorithm 6 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 $\mathbb{S}^n$.

$F \leftarrow np.random.randn(nPts, n + 1)$
$P \leftarrow F/np.linalg.norm(F, axis = 1, keepdims = True)$
return P

Considering Algorithms 1 with linear search of points (Algorithm 2) and constructing α-complex based filtration we got the results showed in Table 1 showing that the number of points per dimension increases in a ratio between $[3, 5]\times$.

Table 1: Results obtained with $n \in [1, 4]$ and considering $nPts$ increment as the older number of points of previous dimension.

n	$nPts$	$ratio\ (nPts_i/nPts_{i-1})$	$confidence$
1	34	–	95
2	170	$5x$	98
3	510	$3x$	96
4	1530	$3x$	97

3 Parallel Approach

In this section, we take advantage of multi-core architectures on Algorithm 4. 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 *"ProcessPoolExecutor"* object from concurrent.futures package. Then, the tasks are submitted asynchronously and processed as soon as finished using *"as_completed"* feature. In the following Python code snipped we summarized the modifications to Algorithm 4.

```
from concurrent.futures import ProcessPoolExecutor, as_completed

def process_points(nPts, n, N, k, ef):
    # failures should not be greater than the ef

    with ProcessPoolExecutor() as executor:
        failures = 0
        items = [(nPts, n, k) for t in range(N)]

        # submit all tasks
        futures = [executor.submit(analyze_persistence, item) for item in items]

        for future in as_completed(futures):
            try:
                if not future.result():
                    failures += 1
                if failures > ef:
                    # as soon as we fail enough we kill the executor
                    # this raises a lot of exceptions due to forcing processes to finish
                    # we redirect those exceptions to par_err.log file
                    for f in futures:
                        f.cancel()                  # Cancel any unfinished tasks
                    executor.shutdown(wait=False)   # cancel remaining tasks immediately

                    break
            except Exception as e:
                failures += 1

    answer = failures <= ef
    return answer, failure
```

After executing the parallel approach the results can be found in Table 2.

Table 2: Results obtained with $n \in [1, 4]$ and considering linear search of points according to Algorithm 2. By the higher confidence obtained it seems that there are a lower number of points that can reach the desired confidence close to 95%.

n	$nPts$	$ratio\ (nPts_i/nPts_{i-1})$	$confidence$
1	34	–	95
2	170	5	100
3	680	3.2	100
4	2040	3	100

Table 3: Results obtained with $n \in [1, 4]$ and considering galloping search of points according to Algorithm 3. Note that we were able to obtain even lower number of points than using the linear search.

n	$nPts$	$ratio\ (nPts_i/nPts_{i-1})$	$confidence$
1	27	–	96
2	145	5	96
3	467	3	96
4	1393	3	96

It should be reviewed why it provides 100 confidence. I stop the computation of the H_5 for 5-sphere after many hours. Moreover, we can see that the ratio of point number per dimension is still between $[3, 5]\times$.

4 Suspension Approach

The suspension $\Sigma(\mathbb{S}^n)$ of an n-sphere provides an alternative approach to understanding the minimal points required for persistent homology in dimension $n + 1$ [5, 3].

The suspension $\Sigma(\mathbb{S}^n)$ of an n-sphere $\mathbb{S}^n$ is formally defined as:

$$\Sigma(\mathbb{S}^n) = (\mathbb{S}^n \times [-1, 1])/\sim$$

where $\sim$ is the equivalence relation that identifies all points $(x, 1)$ to the north pole N and all points $(x, -1)$ to the south pole S.

4.1 Constructing Points on Suspension

If $p = (x_1, \ldots, x_{n+1})$ is a point on $\mathbb{S}^n$:

- Its suspension path would be $(x_1, \ldots, x_{n+1}, t)$ where $t \in [-1, 1]$.
- The poles are $(0, \ldots, 0, \pm 1)$.
- This suggests we need to sample t-values strategically in such a way that their coverage captures the "width" of the suspension.

4.2 Layered Structure Analysis

The suspension structure suggests a natural layered sampling strategy. Let us assume $2m + 3$ layers, where 3 layers correspond to the poles and equator, and m intermediate layers in between each pole and equator.

With this construction, we consider points on the n-sphere come from n-dimensional spherical coordinates [4]:

$$x = \left(\prod_{j=1}^{n} \sin\ \theta_j \right) \cdot \cos\ \theta_{n+1}$$

As it can be noticed, the last coordinate is always a cosine; therefore, the t_i coordinate that we want to add to an n-sphere point corresponds to $\cos(\theta_i)$. Since we are considering $2m + 3$ layers in total, we need the same number of angles to compute the respective heights. We choose $2m + 3$ angles from $[0, \pi]$[1]. Each angle is given by $\{\theta_i = \pi i/(2m + 2)\}_{i\in[0,2m+2]}$.

Recall that in each layer t_i, we have an n-sphere with radius

$$r_i = \sqrt{1 - t_i^2}$$

This is because $x_1^2 + \ldots + x_{n+1}^2 + t_i^2 = r^2 = 1$. Thus, we need to scale the density of points at height t_i to $\#p * r_i$, generate the n-sphere on that layer with $\#p$ points. Since Algorithm 6 samples n-spheres with radius 1, we rescale these points to have radius r_i and attach the coordinate t_i, as summarized by Algorithm 7.

4.3 Implementation details

In this approach, we perform substantial changes:

1. We consider the parallel approach of Section 3 resulting in a modified Algorithm 4.
2. We create a sampling_layered_nsphere algorithm summarized in Algorithm 7, intended to be called in Algorithm 5 instead of *sampling_nsphere*.

Algorithm 7 sampling_layered_nsphere(nPts, n).

Require: A number of points $nPts$ in the equator, the desired dimension of the nsphere n.
Ensure: A collection $P \in \mathbb{R}^{n+1}$ of points in the boundary of $\mathbb{S}^n$.

$heights \leftarrow \{t_i = cos(\theta_i) \mid \theta_i \leftarrow \pi i/(2m+2)\}_{i \in [0,2m+2]}$
$layers \leftarrow \{\}$
for $t_i \in heights$ **do**
  if $t_i = 1$ **then** ▷ North pole
    $layers \leftarrow layers \cup \{(0, \ldots, 1)\}$
  else if $t_i = -1$ **then** ▷ South pole
    $layers \leftarrow layers \cup \{(0, \ldots, -1)\}$
  else ▷ Intermediate
    $r \leftarrow \sqrt{1 - t_i^2}$ ▷ Radius at this height
    $nPts_i \leftarrow \lfloor nPts \cdot r \rfloor$ ▷ Scale density by radius
    $P \leftarrow sampling\_nsphere(nPts_i, n-1)$ ▷ calling Algorithm 6 with dimension $n-1$
    $P' \leftarrow P \cdot r \times \{t_i\}$ ▷ Scale to r and add height as final coordinate
    $layer \leftarrow layer \cup P'$
  end if
end for
return P

We apply the suspension approach during the point generation phase, keeping the rest of the process unchanged. The obtained results are summarized in Table 4. It is worth noting that we successfully computed up to dimension 5 with this approach, achieving a lower number of points per dimension.

The results were obtained using $m = 2$, which corresponds to a total of 7 layers. Interestingly, experimenting with other values of m such as $1, 3, \ldots$ does not necessarily reduce the number of points. However, reducing the k value does result in a lower point count.

Table 4: Results obtained with $n \in [1, 4]$ (using point linear search) with suspension theory to provide a layered n-sphere.

n	**Base** $nPts$	**Total** $nPts$	$ratio\ (nPts_i/nPts_{i-1})$	m **needed**	$confidence$
1	6	23	–	2	100
2	46	171	7	2	100
3	171	639	3.7	2	100
4	639	2385	3.7	2	100
5	2385	8824	3.7	2	100

5 Future Work

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

- **Limitations of the Proposed Approach:** Unfortunately, the proposed approach does not directly address the *minimal number of points* required to create an n-cycle with persistence k. Instead, it estimates *how dense the n-sphere must be* to produce an n-cycle with the desired persistence. A more direct approach would involve explicitly constructing an n-cycle on the n-sphere and reporting the number of points required.

  One idea we considered was designing a *canonical n-cycle*, projecting it onto the n-sphere, and then modifying the resulting p-cycle by adding or removing p-simplices until the desired persistence was achieved. However, we were unable to devise a *practical method* to accomplish this. Furthermore, as noted in [2], the problem of generating *optimal p-cycles* is known to be *NP-hard*, posing significant computational challenges.

- **Inverse Problem and Harmonic Persistent Homology:** Given the recent interest in *harmonic persistent homology* [1], it is worth exploring the *inverse problem*:

  > *"Can we construct a harmonic cycle on the n-sphere and ensure that it uniquely represents a bar with the desired persistence?"*

[1] the vertical semicircle where $x = \sqrt{r^2 - y^2}$

The underlying hypothesis is that a *harmonic representative* of a persistence interval may correspond to an *optimal or near-optimal p-cycle*. Consequently, such a harmonic cycle would likely require a *minimal number of points*, offering a promising avenue for further investigation.

The source code of discussed approaches is available in `https://github.com/rolan2kn/minimal_points_in_nsphere`.

References

[1] Saugata Basu and Nathanael Cox. *Harmonic Persistent Homology*. 2022. arXiv: `2105.15170 [math.AT]`. URL: `https://arxiv.org/abs/2105.15170`.

[2] Tamal Dey and Jusu Wang. *Computational Topology for Data Analysis: Notes from Book*. Accessed: 11-18-2024.

[3] Herbert Edelsbrunner and John Harer. *Computational Topology - an Introduction*. Michigan, USA: American Mathematical Society, 2010. ISBN: 978-0-8218-4925-5. DOI: `10.1007/978-3-540-33259-6_7`.

[4] C. W. J. Granger. "Multidimensional Gaussian distributions, by K. S. Miller, published by John Wiley and Sons, New York, 1964, viii + 129 pages. The SIAM series in Applied Mathematics". In: *Naval Research Logistics Quarterly* 11.2 (June 1964), pp. 231–231. DOI: `10.1002/nav.3800110214`. URL: `https://ideas.repec.org/a/wly/navlog/v11y1964i2p231-231.html`.

[5] Allen Hatcher. *Algebraic topology*. Cambridge: Cambridge Univ. Press, 2000. URL: `https://cds.cern.ch/record/478079`.