A PERSISTENCE LANDSCAPES TOOLBOX FOR TOPOLOGICAL STATISTICS

PETER BUBENIK AND PAWEŁ DŁOTKO

Abstract. Persistence landscapes can be used to analyze large families of large persistence diagrams. In this paper we discuss efficient algorithms and their implementation to compute and manipulate persistence landscapes. We give concrete examples in which persistence landscapes allow calculations that are out of reach of other methods at present. These algorithms are intended to facilitate the use of statistics in topological data analysis.

1. Introduction

One of the main tools in applied topology is persistent homology. It captures the evolution of the homology of a finite filtered complex $\mathcal{K}_0 \subset \mathcal{K}_1 \subset \ldots \mathcal{K}_n$ as the filtration parameter changes. Applying homology with field coefficients to the filtered complex we obtain a sequence of vector spaces and linear maps,

$$H(\mathcal{K}_0) \hookrightarrow H(\mathcal{K}_1) \hookrightarrow \ldots \hookrightarrow H(\mathcal{K}_n). \tag{1.1}$$

We say that a class c is *born* in $H(\mathcal{K}_b)$ if $c \in H(\mathcal{K}_b)$ and c is not in the image of $H(\mathcal{K}_{b-1})$. We say that the class c *dies* in $H(\mathcal{K}_d)$ if the image of c in $H(\mathcal{K}_{d-1})$ is nonzero, but the image of c in $H(\mathcal{K}_d)$ is zero or if c became identical to other class born earlier. The *birth-death* pair (b, d) is represents the lifespan of the class c. The fundamental result of persistent homology is that all of the information in (1.1) can be captured by a finite multiset of birth-death pairs, called either a *barcode* or a *persistence diagram* [1, 2]. There are various programming libraries available for computing birth-death pairs [3, 4, 5, 6].

The standard metrics on the set of persistence barcodes are the bottleneck metric [7] and the Wasserstein metric [8]. The persistence homology is stable in those distances with respect to perturbations of a filtration [7, 8] which makes it a nice tool in data analysis. However these distances are hard to compute. Their computational complexity is $O(n^3)$ where n is the number of birth-death pairs [9].

One of the simplest statistical questions one can ask of a set of persistence diagrams is, what is their average? There is a nice notion of average in a metric space given by the Fréchet mean, however for persistence diagrams it need not be unique. Notable progress has been made in this direction [10, 11, 12], however there still do not exist effective algorithms for computing means for a wide variety of examples.

In [13] the *persistence landscape* is introduced to help with both computational and statistical challenges when working with persistence diagrams. Given a pair of numbers (b, d) with $b < d$, the piecewise linear (PL) function $f_{(b,d)} : \mathbb{R} \to [0, \infty]$ is defined by:

$$f_{(b,d)} = \begin{cases} 0 & \text{if } x \notin (b, d) \\ x - b & \text{if } x \in (b, \frac{b+d}{2}] \\ -x + d & \text{if } x \in (\frac{b+d}{2}, d) \end{cases} \tag{1.2}$$

We remark that this definition makes sense for $-\infty \leq b < d \leq \infty$.

Date: December 4, 2013.

The *persistence landscape* of a multiset of persistence barcodes $\{(b_i, d_i)\}_{i=1}^n$ is a set of functions $\lambda_k : \mathbb{R} \to \mathbb{R}$ such that $\lambda_k(x) =$ k-th largest value of $\{f_{(b_i,d_i)}(x)\}_{i=1}^n$. We assume that $\lambda_k(x) = 0$ is the k-th largest value do not exist. The persistence landscape is the image of a mapping from persistence diagrams to a larger vector space in which efficient algorithms exist to compute L^p distances for $p \in [1, \infty]$ and the average is easily defined.

In [13] it is shown that the persistence landscape is stable with respect to the L^p distance for $1 \leq p \leq \infty$ and that this L^p distance gives lower bounds for the bottleneck and Wasserstein distances of the corresponding persistence diagrams.

In this paper we discuss efficient algorithms and their implementation to compute and manipulate persistent landscapes. In particular, we show that by using persistence landscapes one can accomplished many goals that are beyond reach when using persistence diagrams and Wasserstein or bottleneck distances.

Our main results are algorithms to do the following. Calculating the persistence landscape from a persistence diagram of size n in $O(n^2)$. If the coordinates of the points in the persistence diagram lie on a grid of size m, then we can do this in $O(mn \log(n))$. We can calculate the average persistence landscape for N persistence diagrams in $O(n^2N^2)$. In addition we can calculate the L^p distance for $1 \leq p \leq \infty$ between the average persistence landscapes for two groups of persistence diagrams in $O(n^2N^2)$.

In Section 2 the basic algorithms and data structures needed to store and construct persistence landscapes are given. In the Section 3 we show how to average families of persistence landscapes and calculate the difference between two such averages. In the Section 4 algorithms to compute distances between landscapes are discussed. Finally, in the Section 5 numerical experiments are presented. The main experiment we are calculate the average persistence landscapes of points sampled uniformly from S^d for $d \in \{2, \ldots, 10\}$. We calculate the distances between these average landscapes and show that they are significantly different. In the remaining two examples we discuss an effect of scaling of a sensor network to the process of how the holes are glued and we compare the times needed to compute distances between a family of persistence diagrams.

2. Data structures and algorithms

2.1. **Input.** We will assume that input consists of a list of n pairs of numbers (b, d) with $b < d$. Each of these pairs represents the birth and death times of a persistent homology class. Thinking of these pairs as points we obtain a persistence diagram [9], and considering them to be intervals we obtain a barcode [14].

We will give two algorithms for calculating persistence landscapes. In the second algorithm we make two additional assumptions. First we assume that b and d are finite. This can be achieved by removing or truncating infinite intervals or by using extended persistence [15]. The second assumption is that each b and d is an element of a finite, evenly-spaced grid, $a, a+d, a+2d, \ldots, a+(m-1)d$. An example of the latter is the output of Perseus [17, 3] or Plex [6].

2.2. **Output.** A *persistence landscape* λ is a function $\lambda : \mathbb{R} \times \mathbb{N} \to [0, \infty]$, or equivalently, a sequence of functions $\lambda_k : \mathbb{R} \to [0, \infty]$ where $k \geq 1$. For every fixed k, λ_k is a PL function. Since the input consists of n birth-death points, $\lambda_k = 0$ for k greater than some fixed $K \leq n$. We will represent λ_k by a sorted list $\mathbb{L}_k$ of the points $(x, \lambda_k(x))$ such that λ_k is not differentiable in x. In addition we keep in $\mathbb{L}_k$ the points $(-\infty, 0)$ and $(\infty, 0)$ or $(\pm\infty, \infty)$ if infinite intervals allowed. The projection of $\mathbb{L}_k$ onto its first coordinate is the vector of *critical points* and the projection on the second coordinates is the vector of *critical values*. We will sometimes abuse notation and refer to the elements of $\mathbb{L}_k$ as *critical points*. Clearly, λ_k can be recovered from $\mathbb{L}_k$ by linearly interpolating consecutive points.

2.3. **Algorithms.** In this section we present two algorithms to construct a persistence landscape. Algorithm 1 does not make any assumptions about the input. Its computational complexity is

$O(n\log(n) + nK)$, where n denotes the number of input pairs and K the number of nonzero landscapes. Note that K is bounded by n. Algorithm 2 assumes that the input coordinates are elements of a finite, evenly-spaced grid of size m. Its computational complexity is $O(mn\log(n))$.

In Algorithm 1 we calculate the persistence landscape using an iterative line-sweep algorithm. Each pass of the line sweep calculates the next $\mathbb{L}_k$. Figure 1 illustrates Algorithm 1 for a simple example.

Algorithm 1 Compute the persistence landscape.

Input: $A = \{(b_i, d_i)\}_{i=1}^n$ – list of pairs, $-\infty \le b_i < d_i \le \infty$;
Output: $\{\mathbb{L}_k\}$ – persistence landscape;
  Sort A primarily according to increasing b and secondarily according to decreasing d;
  Let $k = 1$;
  while $A \neq \emptyset$ **do**
    Pop first (b, d) from A remembering its position in A;
    if $(b, d) = (-\infty, \infty)$ **then**
      Add $(-\infty, \infty), (\infty, \infty)$ to $\mathbb{L}_k$; ++k; continue;
    if $d = \infty$ **then**
      Add $(-\infty, 0), (b, 0), (\infty, \infty)$ to $\mathbb{L}_k$; ++k; continue;
    if $b = \infty$ **then**
      Add $(-\infty, \infty)$ to $\mathbb{L}_k$;
    else
      Add $(-\infty, 0), (b, 0), (\frac{b+d}{2}, \frac{d-b}{2})$ to $\mathbb{L}_k$;
    loop
      if d maximal among remaining d's **then**
        Add $(d, 0), (\infty, 0)$ to $\mathbb{L}_k$; ++k; continue;
      Let d' be first of remaining d's with $d' > d$;
      Pop (b', d') from A, remembering its position;
      if $b' > d$ **then**
        Add $(0, d)$ to $\mathbb{L}_k$;
      if $b' \ge d$ **then**
        Add $(0, b')$ to $\mathbb{L}_k$;
      else
        Add $(\frac{b'+d}{2}, \frac{d-b'}{2})$ to $\mathbb{L}_k$;
        Push $(\frac{b'+d}{2}, \frac{d-b'}{2})$ into A in order (requires iteration between positions of (b, d) and (b', d') in A);
      if $d' = \infty$ **then**
        Add (∞, ∞) to $\mathbb{L}_k$; ++k; continue;
      add $(\frac{b'+d'}{2}, \frac{d'-b'}{2})$ to $\mathbb{L}_k$;
      $(b, d) \leftarrow (b', d')$;

Let us discuss the while loop. At the beginning $|A| = n$. When the algorithm runs, points are removed from A and points are added to A. But in order to add a point to A, a point has to be removed from A. Moreover a point is added to A only if the graphs of two functions given by (1.2) intersect with slopes of 1 and -1 respectively. Since for any two such functions there can be at most one such intersection, there are at most $\binom{n}{2}$ such intersections. Therefore Algorithm 1 terminates.

Let us discuss the time complexity of Algorithm 1. Sorting the points takes $O(n\log(n))$ time. Since the points added to A during the construction $\mathbb{L}_k$ do not affect the computation of $\mathbb{L}_k$, a single iteration of the **while** loop takes $O(|A|)$ time, where $|A| \le n$ is the cardinality of A at the start of the loop. Therefore the computational time to compute $\mathbb{L}_k$ will always be bounded by $O(n)$. Therefore the overall complexity of the Algorithm 1 is $O(n\log(n) + Kn)$, where K is the highest number such that $\mathbb{L}_K$ is not empty. Of course K may be n, and then Algorithm 1 has pessimistic complexity $O(n^2)$.

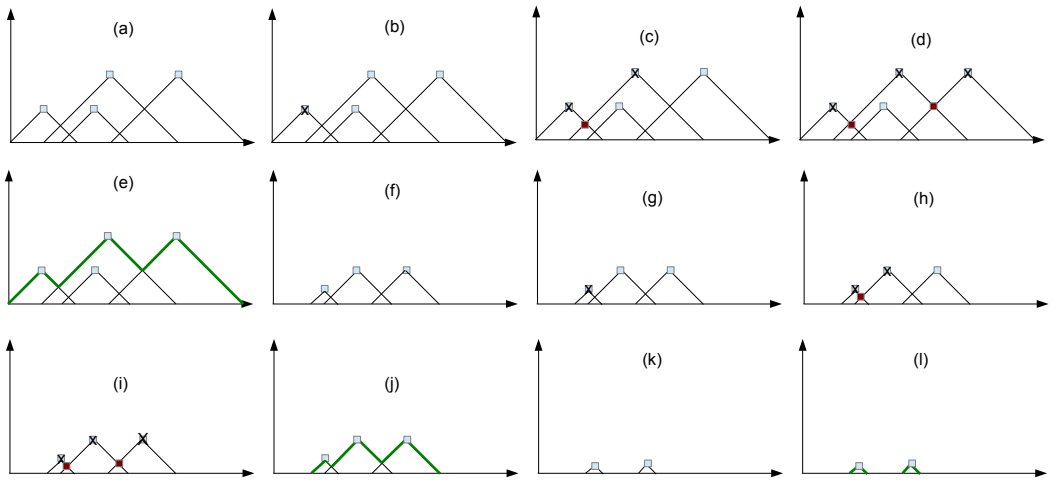

FIGURE 1. Algorithm 1 is used to construct a persistence landscape. (a) The functions (1.2) corresponding to 4 birth-death pairs and their corresponding critical points. (b-d) Steps through the **while** loop to construct $\mathbb{L}_1$. (e) The graph of λ_1. (f) The graph of the functions corresponding to the remaining pairs in the list A. (g-i) The second iteration of the **while** loop constructs $\mathbb{L}_2$. (j) The graph of λ_2. (k) The graphs of the functions corresponding to the remaining pairs on the list A. (l) The graph of λ_3.

Algorithm 2 Compute the persistence landscape using a grid.

Input: $\{(b_i, d_i)\}_{i=1}^n$ – list of pairs $b_i < d_i$ which are a subset of $a, a+d, \ldots, a+(m-1)d$;
Output: $\{\mathbb{L}_k\}$ – persistence landscape;

```
V = vector of the size 2m of (initially empty) vectors of real numbers;
for int i = 1 to n do
    double v = 0, int j = ⌈(b_i − a)/(2d)⌉;
    while v ≤ (d_i − b_i)/2 do
        Add v to V[j];
        v = v + d/2, ++j;
    v = v − d;
    while v ≥ 0 do
        Add v to V[j];
        v = v − d/2, ++j;
for int i = 1 to 2m do
    Sort V[i];
l = max_{i∈{1,...,2m}} size of V[i];
for int i = 0 to l do
    for int k = 0 to 2m do
        if V[k].size() > i then
            Add (a + kd/2, V[k][i]) to L_i;
```

3. AVERAGES

In Section 2 we encoded a persistence landscape $\lambda = \{\lambda_k : \mathbb{R} \to \mathbb{R}\}$ by a list $\mathbb{L}_k$ of pairs of extended real numbers. Define $\mathbb{X}_k$ and $\mathbb{Y}_k$ to be the vectors of critical numbers and critical values

obtained from the first and second coordinates of elements of $\mathbb{L}_k$. Then $\mathbb{Y}_k = \lambda_k(\mathbb{X}_k)$, and λ_k can be obtained from $\mathbb{X}_k$ and $\mathbb{Y}_k$ by linear interpolation.

Now suppose that we have persistence landscapes $\lambda^1, \ldots, \lambda^N$ and we wish to calculate the linear combination $f = \sum_{j=1}^N a_j \lambda^j$, where $a_j \in \mathbb{R}$. Let $f_k(t) = f(k,t)$. Then $f_k = \sum_{j=1}^N a_j \lambda_k^j$. Here we give an algorithm for calculating a representation of f_k from the representations $(\mathbb{X}_k^1, \mathbb{Y}_k^1), \ldots, (\mathbb{X}_k^N, \mathbb{Y}_k^N)$ of $\lambda_k^1, \ldots, \lambda_k^N$.

First we sort the union of the elements of $\mathbb{X}_k^1, \ldots, \mathbb{X}_k^N$, removing repetitions. Call this vector $\mathbb{X}_k$. For each $1 \leq j \leq N$, define $\bar{\mathbb{Y}}_k^j = \lambda_k^j(\mathbb{X}_k)$. Now we represent λ_k^j by $\mathbb{X}_k$ and $\bar{\mathbb{Y}}_k^j$. Again, λ_k^j can be obtained from $\mathbb{X}_k$ and $\bar{\mathbb{Y}}_k^j$ by linear interpolation. By definition $f_k(\mathbb{X}_k) = \sum_{j=1}^N a_j \lambda_k^j(\mathbb{X}_k) = \sum_{j=1}^N a_j \bar{\mathbb{Y}}_k^j$. Also, f_k can be recovered from $\mathbb{X}_k$ and $f_k(\mathbb{X}_k)$ be linear interpolation.

In summary, vector space operations on $\lambda_k^1, \ldots, \lambda_k^N$ are obtained from vector space operations on $\bar{\mathbb{Y}}_k^1, \ldots, \bar{\mathbb{Y}}_k^N$.

Algorithm 3 Linear combination of persistence landscapes.

Input: $(\mathbb{X}_k^1, \mathbb{Y}_k^1), \ldots, (\mathbb{X}_k^N, \mathbb{Y}_k^N)$
Output: $(\mathbb{X}_k, \mathbb{Y}_k)$
Merge the sorted lists $\mathbb{X}_k^j$, removing duplicates. Call this vector $\mathbb{X}_k$.
for $j = 1$ to N **do**
Calculate $\bar{\mathbb{Y}}_k^j = \lambda_k^j(\mathbb{X}_k)$ by linear interpolation.
$\mathbb{Y}_k \leftarrow \sum_{j=1}^N a_j \bar{\mathbb{Y}}_k^j$.
return $(\mathbb{X}_k, \mathbb{Y}_k)$

Let m be the maximum number of the critical points of the persistence landscapes. Then constructing $\mathbb{X}_k$ takes $O(mN \log(mN))$. Since the length of X_k may be at most mN, calculating each $\bar{\mathbb{Y}}_k^j$ takes $O(mN^2)$ and calculating $\mathbb{Y}_k$ takes $O(mN^2)$. So Algorithm 3 has time complexity $O(mN^2)$.

Now if we repeat Algorithm 3 for all k we obtain a linear combination of the full persistence landscape. This has time complexity $O(n^2N^2)$, where n is the maximum number of birth-death pairs generating the persistence landscapes, since there are most n nontrivial $(\mathbb{X}_k, \mathbb{Y}_k)$ and $m \leq n$.

As special cases, we can use this algorithm to calculate the average of a list of persistence landscapes, $\overline{\lambda} = \sum_{j=1}^N \frac{1}{N} \lambda^j$, or the difference between the averages of two groups of persistence landscapes, $\overline{\lambda} - \overline{\lambda'} = \sum_{j=1}^N \frac{1}{N} \lambda^j + \sum_{j=1}^{N'} -\frac{1}{N'} \lambda'^j$.

4. Distances

In this section we consider the computations of the L^p and L^∞ distances between functions f, g that are linear combinations of a set of persistence landscapes. Let $(\mathbb{X}, \mathbb{Y})$ be the representation of $f - g$ as described in Section 3. Let K be the maximum k for which $(\mathbb{X}_k, \mathbb{Y}_k)$ is nontrivial.

The L^∞ distance between f and g is given by the L^∞ norm of the corresponding representation. That is,

$$\left\| f - f' \right\|_\infty = max_k \left\| \mathbb{Y}_k \right\|_\infty .$$

This calculation has time complexity $O(KmN)$.

The L^p distances between f and f' is given by the formula:

$$\left\| f - f' \right\|_p = \sqrt[p]{\sum_{k=1}^{\max(K,K')} \int \left\| f_k - f'_k \right\|_p^p}$$

The norm $\|f_k - f'_k\|_p^p$ can be computed from $(\mathbb{X}_k, \mathbb{Y}_k)$ by summing integrals over intervals given by consecutive elements of $\mathbb{X}_k$. These have the form $\int_c^d |ax+b|^p\, dx = \frac{(ax+b)^{p+1}}{a(p+1)}|_c^d$. This calculation also has time complexity $O(KmN)$.

Now assume that we start with two sets of persistence diagrams each of which has at most n birth-death pairs, and the larger set consists of N persistence diagrams. Combining results from Sections 3 and 4, since $K \leq n$ and $m \leq n$ we can calculate the L^∞ and L^p distances between the corresponding average persistence landscapes in $O(n^2N^2)$.

5. Experiments

The experiments presented here are proof of concept. Their aim is not to solve any of the considered problems, but to show the utility of the concept of persistence landscapes and the described implementation.

5.1. **Points sampled from S^d.** In this experiment we consider the following questions. Suppose we are given a set of points in $\mathbb{R}^d$. How are these points distributed? If they are sampled from a lower dimensional space, can we detect it using persistent homology?

To help address these type of questions we have performed the following computations. We have sampled 100 points randomly from $(d+1)-$dimensional Gaussian distribution for $d \in \{2, \ldots, 10\}$. We have projected those points to S^d. For an example of such normalization in topological data analysis, see [16]. This is equivalent to sampling 100 points from S^d using the uniform distribution. Then, we compute the persistent homology of the Vietoris-Rips complex of this point cloud. Such a computation was repeated 100 times for every dimension. The radius parameter changed form 0 to a radius in which all inessential 0, 1, and 2 dimensional cycles are killed, which is 0.7 for this range of dimensions. The resulting average persistence landscapes for dimension zero, one and two are in Figure 4, Figure 5, and Figure 6, respectively. The distance between average landscapes for various dimensions has been computed. The results in dimension zero are summarized in the Table 1, and for dimension one in the Table 2 and form dimension two in Table 3.

To validate the obtained results we performed a permutation test. For every pair $i, j \in \{2, \ldots, 10\}$ such that $i \neq j$ the two corresponding sets of 100 persistence landscapes are combined in a set A of cardinality 200. Then the set A is randomly split into two subsets A_1, A_2 of cardinality 100 each. Then average landscapes λ_1 and λ_2 are computed based on landscapes in A_1 and A_2, respectively. Let d denote the distance between the original averaged landscapes in dimensions i and j. The distance between λ_1 and λ_2 is compared to d. This described process is repeated 10000 times. The p value equals the proportion of cases in which the distance between λ_1 and λ_2 is greater than d. For every pair $i \neq j$, it never happened that the distance between λ_1 and λ_2 was greater than the corresponding d. Therefore we can conclude that there there is very strong statistical difference between the persistence landscapes in various dimensions.

We hope that the ability to easily perform such calculations will be very useful in topological data analysis.

5.2. **Scaling in planar Rips complexes.** This experiment is motivated by problems in sensor networks. For further information about topological sensor networks consult [18]. For the whole subsection we keep the average number of sensors per unit area fixed. Suppose the sensors are distributed randomly and the coverage radius R is a little higher than the coverage radius needed to provide the coverage with high probability. We are considering the persistence intervals for the radius parameter $r \in [0, R]$. We want to find out if the process in which all one dimensional cycles are glued is depended on a scale or not.

The setup of the experiment is as follow. $130a^2$ sensors are distributed on a square $M_a = [10a, 10a] \times [10a, 10a]$. The radius r used in construction of Rips complex is such that $r \in [0, 2]$ and

0	0.0993	0.1377	0.1595	0.18605	0.209	0.2195	0.2282	0.2365
0.0993	0	0.10695	0.1442	0.1696	0.1919	0.2093	0.2198	0.23035
0.1377	0.10695	0	0.1177	0.1446	0.1658	0.1823	0.1952	0.2071
0.1595	0.1442	0.1177	0	0.087	0.1127	0.1321	0.1447	0.1593
0.18605	0.1696	0.1446	0.087	0	0.0536	0.0727	0.0984	0.1229
0.209	0.1919	0.1658	0.1127	0.0536	0	0.0356	0.0619	0.087
0.2195	0.2093	0.1823	0.1321	0.0727	0.0356	0	0.0292	0.0543
0.2282	0.2198	0.1952	0.1447	0.0984	0.0619	0.0292	0	0.0259
0.2365	0.23035	0.2071	0.1593	0.1229	0.087	0.0543	0.0259	0

TABLE 1. Distance matrix between averaged landscapes in dimension 0 of set of points in S^d for $d \in \{2, \ldots, 10\}$.

0	0.0905	0.0867	0.0792	0.0752	0.0713	0.0678	0.0614	0.0588
0.0905	0	0.0452	0.06275	0.0815	0.0883	0.0899	0.09035	0.09045
0.0867	0.0452	0	0.0331	0.056	0.0722	0.0808	0.0847	0.0865
0.0792	0.06275	0.0331	0	0.0263	0.0447	0.05955	0.0704	0.0757
0.0752	0.0815	0.056	0.0263	0	0.0208	0.0376	0.0506	0.062
0.0713	0.0883	0.0722	0.0447	0.0208	0	0.0179	0.0318	0.0448
0.0678	0.0899	0.0808	0.05955	0.0376	0.0179	0	0.016	0.0291
0.0614	0.09035	0.0847	0.0704	0.0506	0.0318	0.016	0	0.0142
0.0588	0.09045	0.0865	0.0757	0.062	0.0448	0.0291	0.0142	0

TABLE 2. Distance matrix between averaged landscapes in dimension 1 of set of points in S^d for $d \in \{2, \ldots, 10\}$.

0	98.8446	14.433	16.7507	18.2941	19.1773	19.3288	19.8472	20.3933	20.3888
98.8446	0	99.4143	99.969	98.9645	99.2153	99.27	100.799	100.917	98.1056
14.433	99.4143	0	7.41727	12.5082	16.1639	18.5834	20.6395	22.1545	22.354
16.7507	99.969	7.41748	0	7.73501	12.8598	16.8738	19.6588	21.7679	22.6858
18.2941	98.9645	12.5086	7.73569	0	6.28071	11.8002	15.9319	18.9312	20.6962
19.1773	99.2154	16.1647	12.8616	6.28311	0	6.66167	11.7402	15.5133	18.2732
19.3288	99.27	18.5843	16.8759	11.8029	6.66358	0	5.86467	10.1659	13.7775
19.8472	100.799	20.6405	19.6616	15.9349	11.7462	5.86728	0	4.96949	9.9325
20.3933	100.917	22.156	21.7712	18.9351	15.52	10.1716	4.97485	0	6.05756
20.3888	98.1056	22.3555	22.6885	20.6997	18.2815	13.7864	9.93782	6.06143	0

TABLE 3. Distance matrix between averaged landscapes in dimension 2 of set of points in S^d for $d \in \{2, \ldots, 10\}$.

change from 0 with a step 0.2. For every fixed a for 100 times we are sampling the points randomly from M_a, compute persistence, create persistence landscapes and average them. At the end, we compute standard deviation and distance from the mean. The results obtained with persistence landscapes are summarized in the table below:

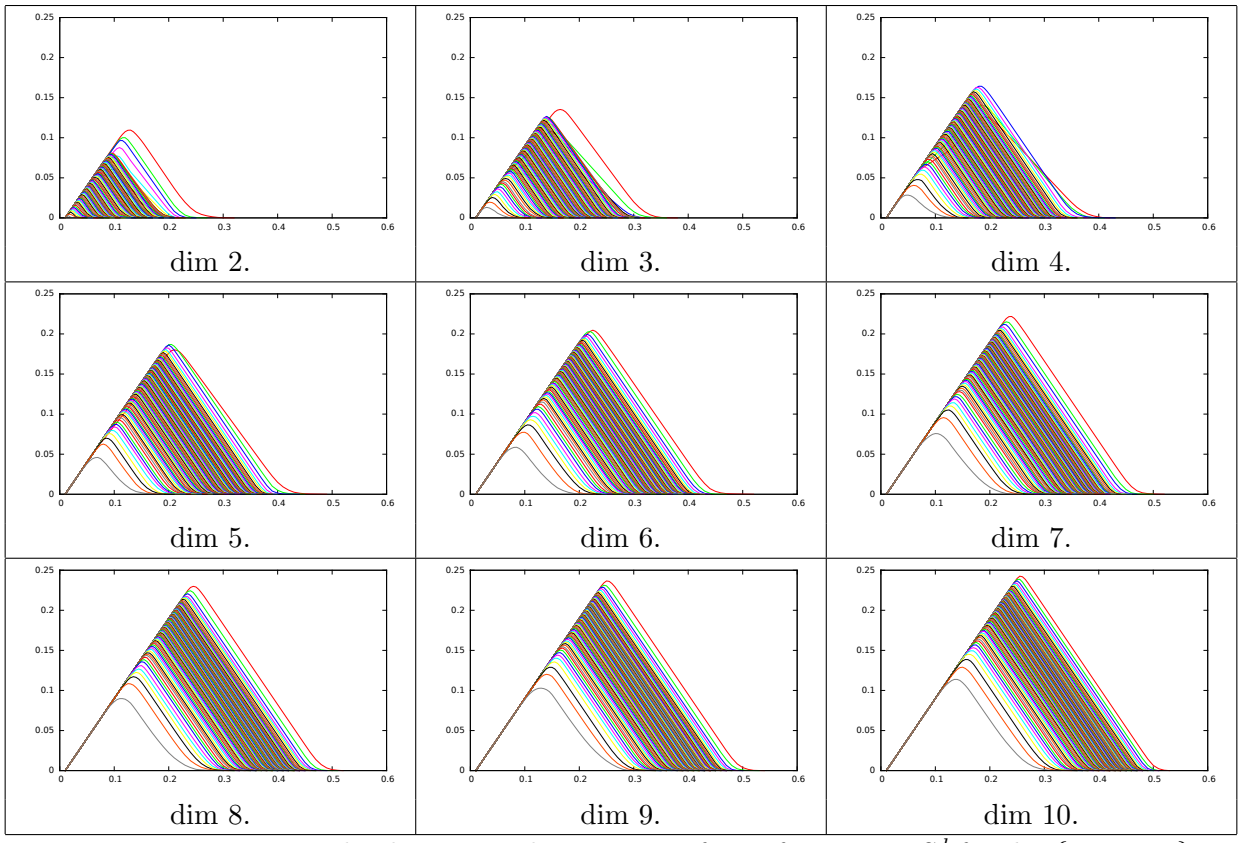

TABLE 4. Persistence landscapes in dimension 0 of set of points in S^d for $d \in \{2, \ldots, 10\}$.

	Dimension 0		Dimension 1	
a	st. deviation	av. distance from average	st. deviation	av. distance from average
1	0.172297	0.16639	0.172285	0.16695
1.5	0.186837	0.17547	0.178868	0.17068
2	0.168297	0.16452	0.183618	0.18167
2.5	0.173492	0.16858	0.188076	0.18494
3	0.17807	0.17346	0.192968	0.18771
3.5	0.17248	0.16868	0.200915	0.1941

The experiments performed are not conclusive in this matter. We were not able to observe any scale dependent pattern in those statistics.

5.3. **Computing distance matrices on a random set of persistence intervals.** In this experiment we have generated a random set of 1400 persistence barcodes of a size ranging from 500 to 700 elements. Our aim is to compute the distance matrix between every pair of persistence barcodes. The computations of the bottleneck distance took tree hours on 512 cores. That gives 1537 hours of computations if they were done on a single core. The computations of the first Wasserstein distance took 27 hours on 512 cores. That gives 13826 on a single core. At the other hand, the computations by using L^∞ and L^1 landscape distance took both less than one hour on a single core. Such a four order of magnitude speedup makes a lot of statistical computations possible

5.4. **Implementation.** The implementation of all the presented procedures is available from the webpage `http://hans.math.upenn.edu/~dlotko/persistenceLandscape.html`.

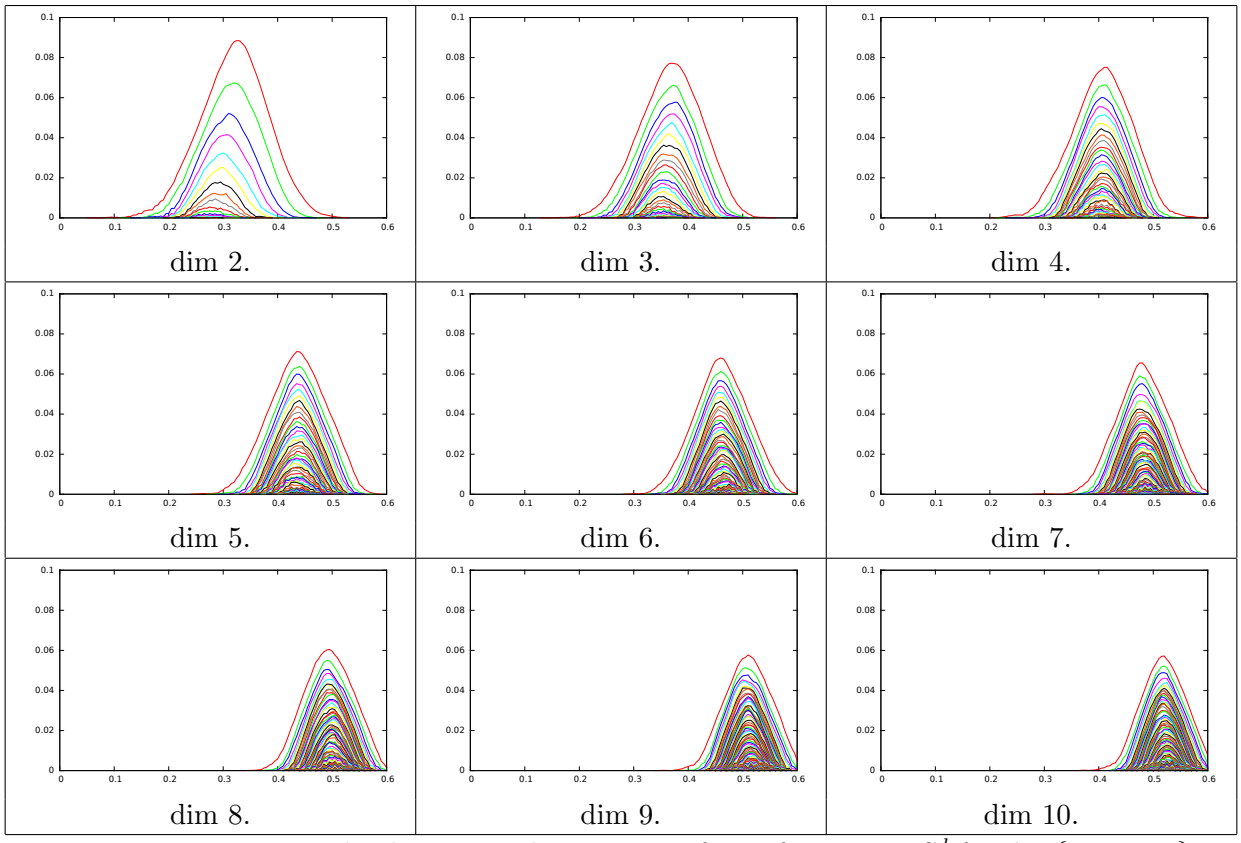

TABLE 5. Persistence landscapes in dimension 1 of set of points in S^d for $d \in \{2, \ldots, 10\}$.

ACKNOWLEDGMENTS

The authors would like to thank Brittany T. Fasy valuable suggestions and Miroslav Kramar for providing timings of a Bottleneck and Wasserstein distance computations. P.B is supported by AFOSR grant FA9550-13-1-0115. P.D is supported by DARPA grant FA9550-12-1-0416 and AFOSR grant FA9550-14-1-0012.

REFERENCES

[1] H. Edelsbrunner, D. Letscher, A. Zomorodian, *Topological persistence and simplification*, Discrete & Computational Geometry, Discrete Comput. Geom. 28 (2002), 511-533.

[2] A. Zomorodian, G. Carlsson, *Computing Persistent Homology*, Discrete & Computational Geometry 33 (2005), 249-274.

[3] V. Nanda, *The Perseus software project*, `www.math.rutgers.edu/~vidit/perseus`, accessed 11/15/2013.

[4] D. Morozov, *The Dionysus software project*, `http://mrzv.org/software/dionysus/`, accessed 11/15/2013.

[5] Ulrich Bauer, Michael Kerber, Jan Reininghaus, *PHAT (Persistent Homology Algorithm Toolbox)*, `https://code.google.com/p/phat/`, accessed 11/15/2013.

[6] *PLEX libdary*, `http://comptop.stanford.edu/u/programs/jplex/`, accessed 11/15/2013.

[7] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, *Stability of Persistence Diagrams*, Discrete & Computational Geometry, Volume 37, Issue 1, pp 103-120 (2007).

[8] D. Cohen-Steiner, H. Edelsbrunner, J. Harer and Y. Mileyko, *Lipschitz functions have L_p-stable persistence*, Found. Comput. Math., 10 (2010), 127-139.

[9] H. Edelsbrunner, J. Harer, *Computational Topology*, American Mathematical Society, 2010.

[10] Y. Mileyko, S. Mukherjee, J. Harer, *Probability measures on the space of persistence diagrams*, Inverse Problems, 27 124007, 2011.

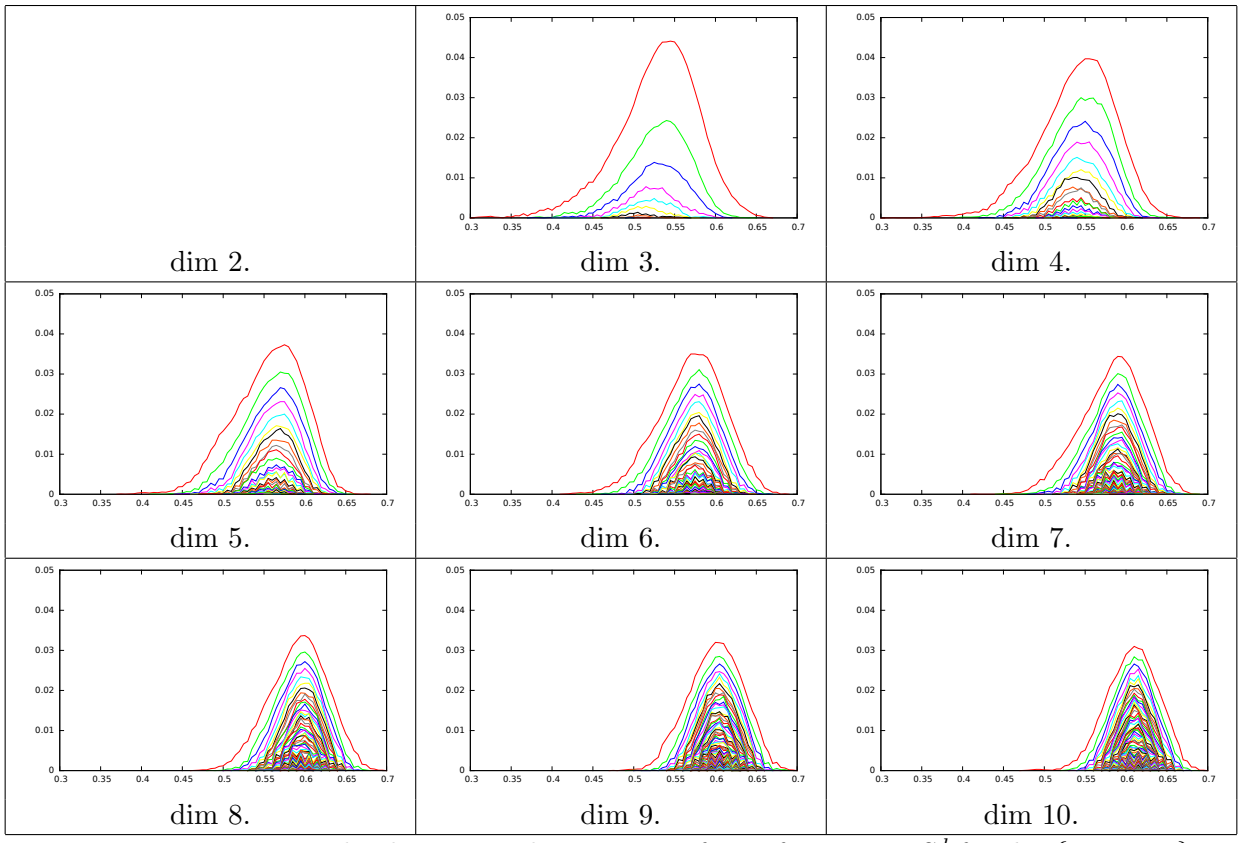

TABLE 6. Persistence landscapes in dimension 2 of set of points in S^d for $d \in \{2, \ldots, 10\}$.

[11] K. Turner, Y. Mileyko, S. Mukherjee, J. Harer, *Fréchet Means for Distributions of Persistence diagrams*, arXiv:1206.2790.

[12] E. Munch, P. Bendich, K. Turner, S. Mukherjee, J. Mattingly, J. Harer, *Probabilistic Fréchet Means and Statistics on Vineyards*, arXiv:1307.6530.

[13] P. Bubenik, *Statistical topology using persistence landscapes*, arXiv:1207.6437.

[14] R. Ghrist, *Barcodes: The persistent topology of data*, Bull. Amer. Math. Soc. 45 (2008), 61-75.

[15] D. Cohen-Steiner, H. Edelsbrunner and J. Harer. *Extending persistence using Poincare and Lefschetz duality*, Found. Comput. Math. 9 (2009), 79-103.

[16] G. Carlsson, T. Ishkhanov, V. de Silva, A. Zomorodian, *On the Local Behavior of Spaces of Natural Images*, Int. J. Comput. Vision, Volume 76, Issue 1, pp 1-12 2008.

[17] K. Mischaikow, V. Nanda, *Morse Theory for Filtrations and Efficient Computation of Persistent Homology*, Discrete & Computational Geometry, September 2013, Volume 50, Issue 2, pp 330-353.

[18] V. de Silva and R. Ghrist, *Coordinate-free coverage in sensor networks with controlled boundaries*, Intl. J. Robotics Research, 25(12), 1205-1222.

DEPARTMENT OF MATHEMATICS, CLEVELAND STATE UNIVERSITY
E-mail address: p.bubenik@csuohio.edu

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF PENNSYLVANIA, PHILADELPHIA, PA AND INSTITUTE OF COMPUTER SCIENCE, JAGIELLONIAN UNIVERSITY, KRAKOW, POLAND
E-mail address: dlotko@sas.upenn.edu