Topological Machine Learning with Persistence Indicator Functions

Bastian Rieck, Filip Sadlo, and Heike Leitte

Abstract Techniques from computational topology, in particular persistent homology, are becoming increasingly relevant for data analysis. Their stable metrics permit the use of many distance-based data analysis methods, such as multidimensional scaling, while providing a firm theoretical ground. Many modern machine learning algorithms, however, are based on kernels. This paper presents *persistence indicator functions* (PIFs), which summarize persistence diagrams, i.e., feature descriptors in topological data analysis. PIFs can be calculated and compared in linear time and have many beneficial properties, such as the availability of a kernel-based similarity measure. We demonstrate their usage in common data analysis scenarios, such as confidence set estimation and classification of complex structured data.

1 Introduction

Persistent homology [9–11], now over a decade old, has proven highly relevant in data analysis. The last years showed that the usage of topological features can lead to an increase in, e.g., classification performance of machine learning algorithms [20]. The central element for data analysis based on persistent homology is the *persistence diagram*, a data structure that essentially stores related critical points (such as minima or maxima) of a function, while providing two stable metrics, namely the *bottleneck distance* and the p^{th} *Wasserstein distance*. Certain stability theorems [7, 8] guarantee that the calculations are robust against perturbations and the inevitable occurrence of noise in real-world data.

B. Rieck (✉) · H. Leitte
TU Kaiserslautern, Kaiserslautern, Germany
e-mail: rieck@cs.uni-kl.de; bastian.rieck@iwr.uni-heidelberg.de; leitte@cs.uni-kl.de

F. Sadlo
Heidelberg University, Heidelberg, Germany
e-mail: sadlo@uni-heidelberg.de

© Springer Nature Switzerland AG 2020

H. Carr et al. (eds.), *Topological Methods in Data Analysis and Visualization V*,
Mathematics and Visualization, https://doi.org/10.1007/978-3-030-43036-8_6

This stability comes at the price of a very high runtime for distance calculations between persistence diagrams: both metrics have a complexity of at least $\mathcal{O}\left(n^{2.5}\right)$, or, if naively implemented, $\mathcal{O}\left(n^3\right)$ [9, p. 196]. Using randomized algorithms, it is possible to achieve a complexity of $\mathcal{O}\left(n^\omega\right)$, where $\omega < 2.38$ denotes the best matrix multiplication time [18]. Further reductions in runtime complexity are possible if approximations to the correct value of the metric are permitted [14]. Nevertheless, these algorithms are hard to implement and their performance is worse than $\mathcal{O}\left(n^2\right)$, meaning that they are not necessarily suitable for comparing larger sets of persistence diagrams.

In this paper, we describe a summarizing function for persistence diagrams, the persistence indicator function (PIF). PIFs were informally introduced in a previous publication [22]. Here, we give a more formal introduction, demonstrate that PIFs can be easily and rapidly calculated, derive several properties that are advantageous for topological data analysis as well as machine learning, and describe example usage scenarios, such as hypothesis testing and classification. We make our implementation, experiments, and data publicly available.[1]

2 Related Work

The *persistence curve* is a precursor to PIFs that is widely used in the analysis of Morse–Smale complexes [4, 13, 17]. It counts the number of certain critical points, such as minima or maxima, that either occur at a given persistence threshold or at given point in time. The curve is then used to determine a relevant threshold, or cut-off parameter for the simplification of the critical points of a function. To the best of our knowledge, no standardized variant of these curves appears to exist.

Recognizing that persistence diagrams can be analyzed at multiple scales as well in order to facilitate hierarchical comparisons, there are some approaches that provide approximations to persistence diagrams based on, e.g., a smoothing parameter. Among these, the stable kernel of Reininghaus et al. [20] is particularly suited for topological machine learning. Another approach by Adams et al. [1] transforms a persistence diagram into a finite-dimensional vector by means of a probability distribution. Both methods require choosing a set of parameters (for kernel computations), while PIFs are fully parameter-free. Moreover, PIFs also permit other applications, such as mean calculations and statistical hypothesis testing, which pure kernel methods cannot provide.

Recently, Bubenik [5] introduced *persistence landscapes*, a functional summary of persistence diagrams. Within his framework, PIFs can be considered to represent a summary (or projection) of the *rank function*. Our definition of PIFs is more straightforward and easier to implement, however. Since PIFs share several properties of persistence landscapes—most importantly the existence of simple

[1] https://github.com/Submanifold/topological-machine-learning.

function-space distance measures—this paper uses similar experimental setups as Bubenik [5] and Chazal et al. [6].

3 Persistence Indicator Functions (PIFs)

Given a persistence diagram $\mathscr{D}$, i.e., a descriptor of the topological activity of a data set [9], we can summarize topological features by calculating an associated persistence indicator function of $\mathscr{D}$ as

$$\begin{aligned} \mathbb{1}_{\mathscr{D}} \colon \mathbb{R} &\longrightarrow \mathbb{N} \\ \epsilon &\longmapsto \left|\left\{(c,d) \in \mathscr{D} \mid \epsilon \in [c,d]\right\}\right|, \end{aligned} \tag{1}$$

i.e., the number of points in the persistence diagram that, when being treated as a closed interval, contain the given parameter ϵ. Equivalently, a PIF can be considered to describe the rank of the p^{th} homology group of a filtration of a simplicial complex. A PIF thus measures the amount of topological activity as a function of the threshold parameter ϵ. This parameter is commonly treated as the "range" of a function defined *on* the given data set, e.g., a distance function [11] or an elevation function [2]. Figure 1 demonstrates how to calculate the persistence indicator function $\mathbb{1}_{\mathscr{D}}(\cdot)$ from a persistence diagram or, equivalently, from a persistence barcode. For the latter, the calculation becomes particularly easy. In the barcode, one only has to check the number of intervals that are intersected at any given time by a vertical line for some value of ϵ.

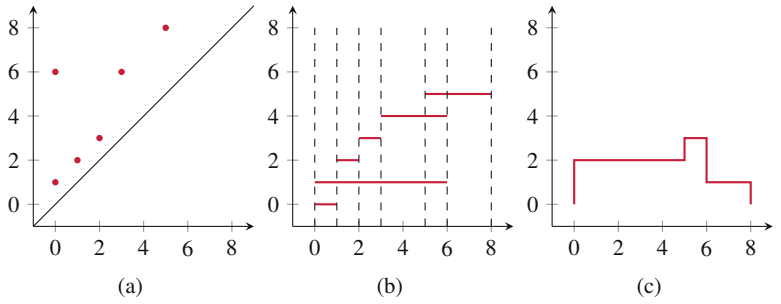


Fig. 1 A persistence diagram (**a**), its persistence barcode (**b**), and its corresponding persistence indicator function (**c**). Please note that the interpretation of the axes changes for each plot

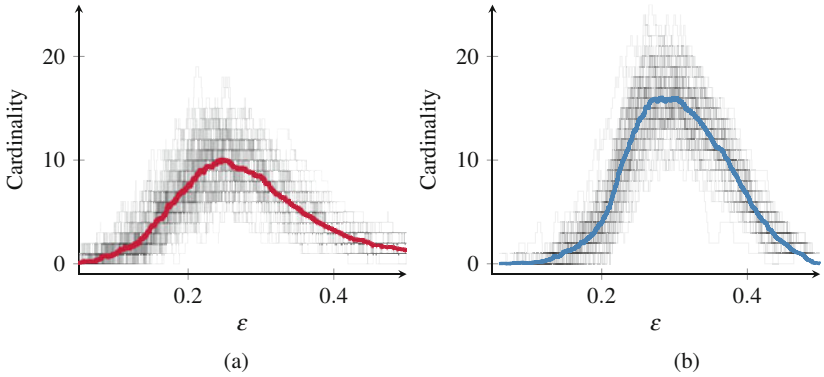


Fig. 2 Mean persistence indicator function of the one-dimensional persistence diagrams of a sphere and of a torus. Both data sets have been sampled at random and are scaled such that their volume is the same. (**a**) Sphere ($r \approx 0.63$). (**b**) Torus ($R = 0.025$, $r = 0.05$)

3.1 Properties

We first observe that the PIF only changes at finitely many points. These are given by the creation and destruction times, i.e., the x- and y-coordinates, of points in the persistence diagram. The PIF may thus be written as a sum of appropriately scaled *indicator functions* (hence the name) of the form $\mathbb{1}_{\mathscr{I}}(\cdot)$ for some interval $\mathscr{I}$. Within the interval $\mathscr{I}$, the value of $\mathbb{1}_{\mathscr{D}}(\cdot)$ does *not* change. Hence, the PIF is a *step function*. Since step functions are compatible with addition and scalar multiplication, PIFs form a vector space. The addition of two PIFs corresponds to calculating the union of their corresponding persistence diagrams, while taking multiplicities of points into account. As a consequence, we can calculate the *mean* of a set of PIFs $\{\mathbb{1}^1_{\mathscr{D}}, \ldots, \mathbb{1}^n_{\mathscr{D}}\}$ as

$$\overline{\mathbb{1}_{\mathscr{D}}}(\cdot) := \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}^i_{\mathscr{D}}(\cdot), \tag{2}$$

i.e., the standard pointwise mean of set of elements. In contrast to persistence diagrams, for which a mean is not uniquely defined and hard to calculate [26], this calculation only involves addition (union) and scalar multiplications of sets of intervals. Figure 2 depicts mean persistence indicator functions for two randomly-sampled data sets. We see that the resulting mean persistence indicator functions already introduce a visual separation between the two data sets.

As a second derived property, we note that the absolute value of a step function (and that of a PIF) always exists; one just calculates the absolute value for every interval in which the step function does *not* change. The absolute value of

a step function is again a step function, so the Riemann integral of a PIF is well-defined, giving rise to their 1-norm as

$$\|\mathbb{1}_{\mathscr{D}}\|_1 := \int_{\mathbb{R}} |\mathbb{1}_{\mathscr{D}}(x)| \mathrm{d}x, \tag{3}$$

which is just the standard norm of an L^1-space. The preceding equation requires the use of an absolute value because linear operations on PIFs may result in negative values. The integral of a PIF (or its absolute value) decomposes into a sum of integrals of individual step functions, defined over some interval $[a, b]$. Letting the value of the step function over this interval be l, the integral of the step function is given as $l \cdot |b-a|$, i.e., the volume of the interval scaled by the value the step function assumes on it. We can also extend this norm to that of an L-space, where $p \in \mathbb{R}$. To do so, we observe that the p^{th} power of any step function is well-defined—we just raise the value it takes to the p^{th} power. Consequently, the p^{th} power of a PIF is also well-defined and we define

$$\|\mathbb{1}_{\mathscr{D}}\|_p := \Big(\int_{\mathbb{R}} |\mathbb{1}_{\mathscr{D}}(x)|^p \mathrm{d}x \Big)^{\frac{1}{p}}, \tag{4}$$

which is the standard norm of an L-space. Calculating this integral again involves decomposing the range of the PIF into individual step functions and calculating their integral. We have $\|\mathbb{1}_{\mathscr{D}}\|_p < \infty$ for every $p \in \mathbb{R}$ because there are only finitely many points in a persistence diagram, so the integrals of the individual step functions involved in the norm calculation are bounded.

Hypothesis Testing Treating the norm of a PIF as a random variable, we can perform topology-based hypothesis testing similar to persistence landscapes [5]. Given two different samples of persistence diagrams, $\{\mathscr{D}_1^1, \ldots, \mathscr{D}_n^1\}$ and $\{\mathscr{D}_1^2, \ldots, \mathscr{D}_n^2\}$, we calculate the 1-norm of their respective mean PIFs as Y_1 and Y_2, and the variances

$$\sigma_i^2 := \frac{1}{n-1} \sum_{j=1}^{n} \Big(\big| \mathbb{1}_{\mathscr{D}_j^i} \big| - Y_i \Big)^2, \tag{5}$$

for $i \in \{1, 2\}$. We may then perform a standard two-sample z-test to check whether the two means are likely to be the same. To this end, we calculate the z-score as

$$z := \frac{Y_1 - Y_2}{\sqrt{s_1^2/n - s_2^2/n}} \tag{6}$$

and determine the critical values at the desired α-level, i.e., the significance level, from the quantiles of a normal distribution. If z is outside the interval spanned by the critical values, we *reject* the null hypothesis, i.e., we consider the two means to be different. For the example shown in Fig. 2, we obtain $Y_1 \approx 2.135$, $s_1^2 \approx 0.074$,

$Y_2 \approx 2.79$, and $s_2^2 \approx 0.093$. Using $\alpha = 0.01$, the critical values are given by $z_1 \approx -2.58$ and $z_2 \approx 2.58$. Since $z \approx -11.09$, we *reject* the null hypothesis with $p \approx 1.44 \times 10^{-28} \ll 0.01$. Hence, PIFs can be used to confidently discern random samples of a sphere from those of a torus with the same volume.

Stability The 1-norm of a PIF is connected to the *total persistence* [8], i.e., the sum of all persistence values in a persistence diagram. We have

$$\|\mathbb{1}_{\mathscr{D}}\|_1 = \int_{\mathbb{R}} |\mathbb{1}_{\mathscr{D}}(x)| \mathrm{d}x = \sum_{I \in \mathscr{I}} c_I \operatorname{vol}(I), \tag{7}$$

where $\mathscr{I}$ denotes a set of intervals for which the number of active persistence pairs does not change, and c_I denotes their count. We may calculate this partition from a persistence barcode, as shown in Fig. 1b, by going over all the dotted slices, i.e., the intervals between pairs of start and endpoints of each interval. The sum over all these volumes is equal to the total persistence of the set of intervals, because we can split up the volume calculation of a single interval over many sub-interval and, in total, the volume of every interval is only accumulated once. Hence,

$$\|\mathbb{1}_{\mathscr{D}}\|_1 \int_{\mathbb{R}} |\mathbb{1}_{\mathscr{D}}(x)| \mathrm{d}x = \sum_{(c,d) \in \mathscr{D}} |d - c| = \sum_{(c,d) \in \mathscr{D}} \operatorname{pers}(c, d) = \operatorname{pers}(\mathscr{D}), \tag{8}$$

where $\operatorname{pers}(\mathscr{D})$ denotes the total persistence of the persistence diagram. According to a stability theorem by Cohen-Steiner et al. [8], the 1-norm of a PIF is thus stable with respect to small perturbations. We leave the derivation of a similar equation for the general L-norm of a PIF for future work.

3.2 The Bootstrap for Persistence Indicator Functions

Developed by Efron and Tibshirani [12], the bootstrap is a general statistical method for—among other applications—computing confidence intervals. We give a quick and cursory introduction before showing how this method applies to persistence indicator functions; please refer to Chazal et al. [6] for more details.

Assume that we have a set of independent and identically distributed variables $X_1, \ldots, X_n$, and we want to estimate a real-valued parameter θ that corresponds to their distribution. Typically, we may estimate θ using a statistic $\widehat{\theta} := s(X_1, \ldots, X_n)$, i.e., some function of the data. A common example is to use θ as the population mean, while $\widehat{\theta}$ is the sample mean. If we want to calculate *confidence intervals* for our estimate $\widehat{\theta}$, we require the distribution of the difference $\theta - \widehat{\theta}$. This distribution, however, depends on the unknown distribution of the variables, so we have to approximate it using an empirical distribution. Let $X_1^*, \ldots, X_n^*$ be a sample of the original variables, drawn with replacement. We can calculate $\widehat{\theta}^* := s(X_1^*, \ldots, X_n^*)$

and approximate the unknown distribution by the empirical distribution of $\widehat{\theta} - \widehat{\theta}^*$, which, even though it is not computable analytically, can be approximated by repeating the sampling procedure a sufficient number of times. The quantiles of the approximated distribution may then be used to construct confidence intervals, leading to the following method:

1. Calculate an estimate of θ from the input data using $\widehat{\theta} := s(X_1, \ldots, X_n)$.
2. Obtain $X_1^*, \ldots, X_n^*$ (sample *with* replacement) and calculate $\widehat{\theta}^* := s(X_1^*, \ldots, X_n^*)$.
3. Repeat the previous step B times to obtain $\widehat{\theta}_1^*, \ldots, \widehat{\theta}_B^*$.
4. Given α, compute an approximate $(1 - 2\alpha)$ quantile interval as

$$[\widehat{\theta}_1, \widehat{\theta}_2] \approx [\widehat{\theta}_B^{*(\alpha)}, \widehat{\theta}_B^{*(1-\alpha)}], \tag{9}$$

   where $\widehat{\theta}_B^{*(\alpha)}$ refers to the α^{th} empirical quantile of the bootstrap replicates from the previous step.

This procedure yields both a lower bound and an upper bound for $\widehat{\theta}$. It is also referred to as the percentile interval for bootstraps [12, pp. 168–176]. More complicated versions—yielding "tighter" confidence intervals—of this procedure exist, but the basic method of sampling with replacement and computing the statistic $s(\cdot)$ on the bootstrap samples remains the same.

In order to apply the bootstrap to *functions*, we require empirical processes [16]. The goal is to find a *confidence band* for a function $f(x)$, i.e., a pair of functions $l(x)$ and $u(x)$ such that the probability that $f(x) \in [l(x), u(x)]$ for $x \in \mathbb{R}$ is at least $1 - \alpha$. Given a function f, let $Pf := \int f \mathrm{d}P$ and $P_n f := n^{-1} \sum_{i=1}^n f(X_i)$. We obtain a *bootstrap empirical process* as

$$\{\mathbb{P} f\}_{f \in \mathscr{F}} := \{\sqrt{n} \left(P_n^* f - P_n f\right)\}, \tag{10}$$

where $P_n^* := n^{-1} \sum_{i=1}^n f(X_i^*)$ is defined on the bootstrap samples (as introduced above). Given the convergence of this empirical process, we may calculate

$$\widehat{\theta} := \sup_{f \in \mathscr{F}} |\mathbb{P} f|, \tag{11}$$

which yields a statistic to use for the bootstrap as defined above. From the corresponding quantile, we ultimately obtain $[\widehat{\theta}_1, \widehat{\theta}_2]$ and calculate a confidence band

$$C_n(f) := \left[P_n f - \frac{\widehat{\theta}_1}{n}, P_n f + \frac{\widehat{\theta}_2}{n} \right] \tag{12}$$

for the empirical mean of a set of PIFs. Figure 3 depicts an example of confidence bands for the mean PIF of the sphere and torus samples. We can see that the confidence band for the torus is extremely tight for $\epsilon \in [0.2, 0.3]$, indicating that

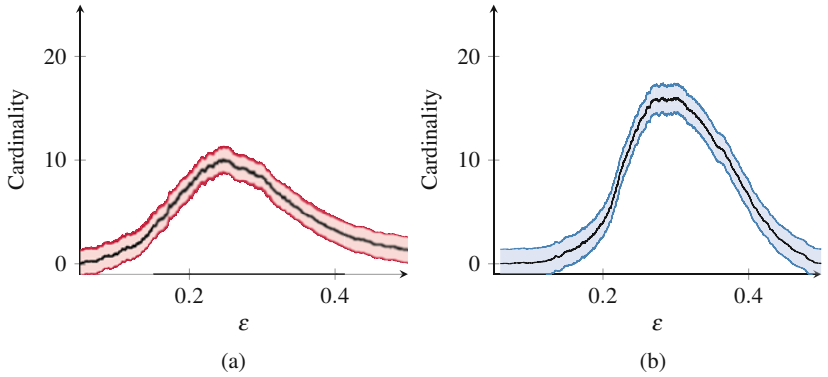


Fig. 3 Confidence bands at the $\alpha = 0.05$ level for the mean persistence indicator functions of a sphere and a torus. The confidence band is somewhat tighter for $\epsilon \geq 0.2$. (**a**) Sphere. (**b**) Torus

the limit behavior of samples from a torus is different at this scale from the limit behavior of samples from a sphere.

3.3 Distances and Kernels

Given two persistence diagrams $\mathscr{D}_i$ and $\mathscr{D}_j$, we are often interested in their dissimilarity (or distance). Having seen that linear combinations and norms of PIFs are well-defined, we can define a family of distances as

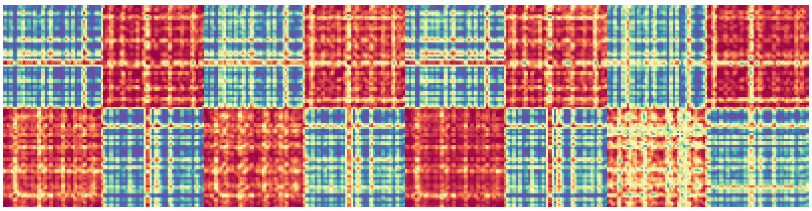
$$\operatorname{dist}_p(\mathbb{1}_{\mathscr{D}_i}, \mathbb{1}_{\mathscr{D}_j}) := \Big(\int_{\mathbb{R}} |\mathbb{1}_{\mathscr{D}_i}(x) - \mathbb{1}_{\mathscr{D}_j}(x)|^p \mathrm{d}x \Big)^{\frac{1}{p}}, \tag{13}$$

with $p \in \mathbb{R}$. Since the norm of a PIF is well-defined, this expression is a metric in the mathematical sense. Note that its calculation requires essentially only evaluating all individual step functions of the difference of the two PIFs once. Hence, its complexity is *linear* in the number of sub-intervals.

Example Figure 4 depicts pairwise distance matrices for random samples of a sphere and of a torus. The first matrix of each group is obtained via dist_p for PIFs, while the second matrix in each group is obtained by the p^{th} Wasserstein distance. We observe two groups of data sets in all matrices, meaning that both classes of distances are suitable for detecting differences.

We can also employ the distance defined above to obtain a *kernel* [23]. To this end, we define

$$k_p(\mathscr{D}_i, \mathscr{D}_j) := -\operatorname{dist}_p(\mathbb{1}_{\mathscr{D}_i}, \mathbb{1}_{\mathscr{D}_j}), \tag{14}$$



(a) (b)

Fig. 4 A comparison of distance matrices obtained using the distance measure dist_p for PIFs, and the corresponding p^{th} Wasserstein distance W_p. The left matrix of each group shows dist_p, while the right matrix depicts W_p. Red indicates close objects (small distances), while blue indicates far objects (large distances). (**a**) $p = 1$. (**b**) $p = 2$

where $p \in \{1, 2\}$ because we need to make sure that the kernel is *conditionally positive definite* [23]. This kernel permits using PIFs with many modern machine learning algorithms. As an illustrative example, we will use *kernel support vector machines* [23] to separate random samples of a sphere and a torus.

Example Again, we use 100 random samples (50 per class) from a sphere and a torus. We only use one-dimensional persistence diagrams, from which we calculate PIFs, from which we then obtain pairwise kernel matrices using both k_1 and k_2. Finally, we train a support vector machine using nested stratified 5-fold cross-validation. With k_1, we obtain an average accuracy of 0.98 ± 0.049, whereas with k_2, we obtain an average accuracy of 0.95 ± 0.063. The decrease in accuracy is caused by the additional smoothing introduced in this kernel.

4 Applications

In the following, we briefly discuss some potential application scenarios for PIFs. We use only data sets that are openly available in order to make our results comparable. For all machine learning methods, we use SCIKIT-LEARN [19].

4.1 Analysis of Random Complexes

It is often useful to know to what extent a data set exhibits random fluctuations. To this end, we sampled 100 points from a unit cube in $\mathbb{R}^3$, which has the topology of a point, i.e., no essential topological features in dimensions >0. We calculated the Vietoris–Rips complex at a scale such that no essential topological features remain in dimensions ≥ 1, and obtained PIFs, which are depicted in Fig. 5 along with their corresponding mean. All functions use a common axis in order to simplify their

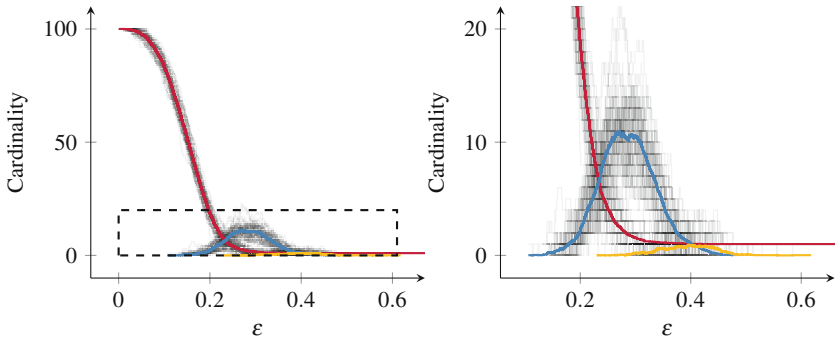


Fig. 5 PIFs for random complexes sampled over a unit cube in $\mathbb{R}^3$. Dimensions zero (red), one (blue), and two (yellow) are shown. To show the peak in dimension two better, the right-hand side shows a "zoomed" version of the first chart (dashed region)

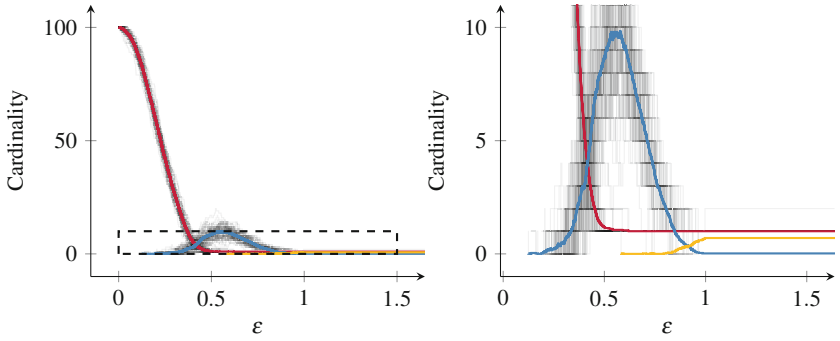


Fig. 6 PIFs for random samples of a sphere with $r = 1.0$. Again, dimensions zero (red), one (blue), and two (yellow) are shown, along with a "zoomed" version of the first chart (dashed region)

comparison. We first comment on the dynamics of these data. Topological activity is "shifted", meaning that topological features with a different dimensionality are not active at the same scale. The maximum of topological activity in dimension one (blue curve) is only reached when there are few zero-dimensional features. The maximum in dimension two (yellow curve) also does not coincide with the maximum in dimension one. These results are consistent with a limit theorem of Bobrowski and Kahle [3], who showed that (persistent) Betti numbers follow a Poisson distribution.

By contrast, for a data set with a well-defined topological structure, such as a 2-sphere, the PIFs exhibit a different behavior. Figure 6 depicts all PIFs of random samples from a sphere. We did not calculate the Vietoris–Rips complex for all possible values of ϵ but rather selected an ϵ that is sufficiently large to capture the correct Betti numbers of the sphere. Here, we observe that the stabilization of topological activity in dimension zero roughly coincides with the maximum

of topological activity in dimension one. Topological activity in dimension two only starts to increase for larger scales, staying stable for a long interval. This activity corresponds to the two-dimensional void of the 2-sphere that we detect using persistent homology.

PIFs can thus be used to perform a test for "topological randomness" in real-world data. This is useful for deciding whether a topological approximation is suitable or needs to be changed (e.g., by calculating a different triangulation, using α-shapes, etc.). Moreover, we can use a PIF to detect the presence or absence of a shared "scale" in data sets. For the random complexes, there is *no* value for ϵ in which stable topological activity occurs in more than one dimension, whereas for the sphere, we observe a stabilization starting from $\epsilon \approx 0.75$.

4.2 Shakespearean Co-occurrence Networks

In a previous work, co-occurrence networks from a machine-readable corpus of Shakespeare's plays [21] have been extracted. Their topological structure under various aspects has been analyzed, using, for example, their clique communities [22] to calculate two-dimensional embeddings of the individual networks. The authors observed that comedies form clusters in these embeddings, which indicates that they are more similar to each other than to plays of another category. Here, we want to show that it is possible to differentiate between comedies and non-comedies by using the kernel induced by PIFs. Among the 37 available plays, 17 are comedies, giving us a baseline probability of 0.46 if we merely "guess" the class label of a play. Since the number of plays is not sufficiently large to warrant a split into test and training data, we use various cross-validation techniques, such as *leave-one-out*. The reader is referred to Kohavi [15] for more details. Table 1 reports all results; we observe that k_1 outperforms k_2. Since k_2 emphasizes small-scale differences, the number of topological features in two networks that are to be compared should be roughly equal. This is *not* the case for most of the comedies, though. We stress that these results are only a demonstration of the capabilities of PIFs; the comparatively low accuracy is partially due to the fact that networks were extracted automatically. It is interesting to note *which* plays tend to be mislabeled. For k_1, ALL'S WELL THAT ENDS WELL, CYMBELINE, and THE WINTER'S TALE are mislabeled more than all other plays. This is consistent with research by Shakespeare scholars who suggest different categorization schemes for these (and other) *problem plays*.

Table 1 Classifier performance for Shakespearean co-occurrence networks

Kernel	5-fold	LOO	LPO ($p = 2$)	LPO ($p = 3$)	LPO ($p = 4$)	Split
k_1	0.84	0.83	0.83	0.80	0.79	0.80
k_2	0.64	0.00	0.67	0.68	0.68	0.68

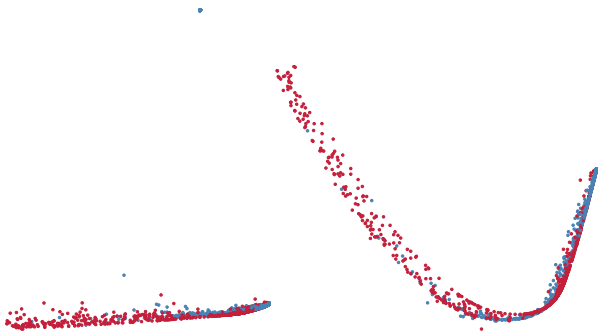
Classification based on k_1 outperforms the second kernel k_2

4.3 Social Networks

Yanardag and Vishwanathan [27] crawled the popular social news aggregation website reddit.com in order to obtain a set of graphs from online discussions. In each graph, the nodes correspond to users and an edge signifies that a certain user responded to a another user's comment. The graphs are partitioned into two classes, one of them representing discussion-based forums (in which users typically communicate freely among each other), the other representing communities based on a question–answer format (in which users typically only respond to the creator of a topic). The data set is fully-balanced.

Here, we want to find out whether it is possible to classify the graphs using nothing but topological information. We use the *degree*, i.e., the number of neighbors, of a node in the graph to obtain a filtration, assigning every edge the maximum of the degrees of its endpoints. We then calculate one-dimensional persistent homology and our kernel for $p = 1$ and $p = 2$. Figure 7 shows the results of applying *kernel principal component analysis* (k-PCA) [24], which each point in the embedding corresponding to a single graph. A small set of outliers appears to "skew" the embedding (Fig. 7a), but an inspection of the data shows that these graphs are extremely small (and sparse) in contrast to the remaining graphs. After removing them, the separation between both classes is visibly better (Fig. 7b). Progress from "left" to "right" in the embedding, graphs tend to become more dense.

As a second application on these data, we use a kernel support vector machine to classify all graphs, without performing outlier removal. We split the data into training (90%) and test (10%) data, and use 4-fold cross validation to find the best hyperparameters. The average accuracy for k_1 is 0.88, while the average accuracy for k_2 is 0.81. PIFs thus manage to surpass previous results by Yanardag and Vishwanathan [27], which employed a computationally more expensive strategy,



(a) Original data (b) Cleaned data

Fig. 7 Embeddings based on k-PCA for the k_1 kernel. (**a**) Every node represents a certain graph. The color indicates a graph from a discussion-based forum (red) or a Q/A forum (blue). (**b**) We removed some outliers to obtain a cleaner output. It is readily visible that the two classes suffer from overlaps, which influence classification performance negatively

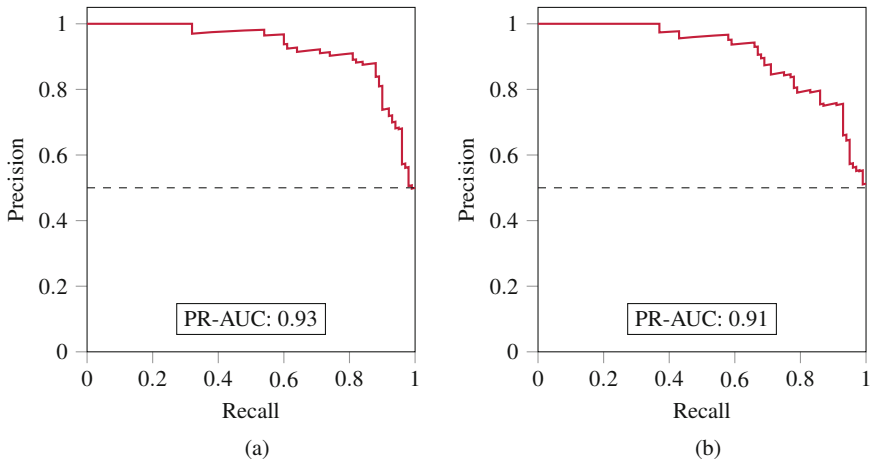


Fig. 8 Precision–recall curves for both kernels on the social networks data set. Each curve also includes the area-under-the-curve (AUC) value. (**a**) k_1. (**b**) k_2

i.e., graph kernels [25] based on learned latent sub-structures, and obtained an average accuracy of 0.78 for these data. Figure 8 depicts precision–recall curves for the two kernels. The kernel k_1 manages to retain higher precision at higher values of recall than k_2, which is again due to its lack of smoothing.

5 Conclusion

This paper introduced persistence indicator functions (PIFs), a novel class of summarizing functions for persistence diagrams. While being approximative by nature, we demonstrated that they exhibit beneficial properties for data analysis, such as the possibility to perform bootstrap experiments, calculate distances, and use kernel-based machine learning methods. We tested the performance on various data sets and illustrated the potential of PIFs for topological data analysis and topological machine learning. In the future, we want to perform a more in-depth analysis of the mathematical structure of PIFs, including detailed stability theorems, approximation guarantees, and a description of their statistical properties.

References

1. Adams, H., Emerson, T., Kirby, M., Neville, R., Peterson, C., Shipman, P., Chepushtanova, S., Hanson, E., Motta, F., Ziegelmeier, L.: Persistence images: a stable vector representation of persistent homology. J. Mach. Learn. Res. **18**(8), 1–35 (2017)
2. Agarwal, P.K., Edelsbrunner, H., Harer, J., Wang, Y.: Extreme elevation on a 2-manifold. Discr. Comput. Geom. **36**(4), 553–572 (2006)
3. Bobrowski, O., Kahle, M.: Topology of random geometric complexes: a survey (2014). https://arxiv.org/abs/1409.4734
4. Bremer, P.T., Edelsbrunner, H., Hamann, B., Pascucci, V.: A topological hierarchy for functions on triangulated surfaces. IEEE Trans. Vis. Comput. Graph. **10**(4), 385–396 (2004). https://doi.org/10.1109/TVCG.2004.3
5. Bubenik, P.: Statistical topological data analysis using persistence landscapes. J. Mach. Learn. Res. **16**, 77–102 (2015)
6. Chazal, F., Fasy, B.T., Lecci, F., Rinaldo, A., Singh, A., Wasserman, L.: On the bootstrap for persistence diagrams and landscapes. Model. Anal. Inf. Syst. **20**(6), 111–120 (2013)
7. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. Discr. Comput. Geom. **37**(1), 103–120 (2007)
8. Cohen-Steiner, D., Edelsbrunner, H., Harer, J., Mileyko, Y.: Lipschitz functions have L_p-stable persistence. Found. Comput. Math. **10**(2), 127–139 (2010)
9. Edelsbrunner, H., Harer, J.: Computational Topology: An Introduction. AMS, New York (2010)
10. Edelsbrunner, H., Morozov, D.: Persistent homology: theory and practice. In: European Congress of Mathematics. EMS Publishing House, Zürich (2014)
11. Edelsbrunner, H., Letscher, D., Zomorodian, A.J.: Topological persistence and simplification. Discr. Comput. Geom. **28**(4), 511–533 (2002)
12. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Monographs on Statistics and Applied Probability, vol. 57 . Chapman & Hall/CRC, Boca Raton, FL (1993)
13. Günther, D., Boto, R.A., Contreras-Garcia, J., Piquemal, J.P., Tierny, J.: Characterizing molecular interactions in chemical systems. IEEE Trans. Vis. Comp. Graph. **20**(12), 2476–2485 (2014)
14. Kerber, M., Morozov, D., Nigmetov, A.: Geometry helps to compare persistence diagrams. In: Goodrich, M., Mitzenmacher, M. (eds.) Proceedings of the 18th Workshop on Algorithm Engineering and Experiments (ALENEX), pp. 103–112. SIAM, Philadelphia, PA (2016)
15. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the IJCAI, vol. 2, pp. 1137–1143 (1995)
16. Kosorok, M.R.: Introduction to Empirical Processes and Semiparametric Inference. Springer, New York, NY (2008)
17. Laney, D., Bremer, P.T., Mascarenhas, A., Miller, P., Pascucci, V.: Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. IEEE Trans. Vis. Comput. Graph. **12**(5), 1053–1060 (2006)
18. Mucha, M., Sankowski, P.: Maximum matchings via Gaussian elimination. In: 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 248–255 (2004)
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É.: SCIKIT-LEARN: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
20. Reininghaus, J., Huber, S., Bauer, U., Kwitt, R.: A stable multi-scale kernel for topological machine learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4741–4748. Curran Associates, Inc., Red Hook, NY (2015)
21. Rieck, B., Leitte, H.: Shall I compare thee to a network?—Visualizing the topological structure of Shakespeare's plays. In: Workshop on Visualization for the Digital Humanities at IEEE VIS. Baltimore, MD (2016)

22. Rieck, B., Fugacci, U., Lukasczyk, J., Leitte, H.: Clique community persistence: a topological visual analysis approach for complex networks. IEEE Trans. Vis. Comput. Graph. **22**(1), 822-831 (2018). https://doi.org/10.1109/TVCG.2017.2744321
23. Schölkopf, B., Smola, A.J.: Learning with Kernels. The MIT Press, Cambridge, MA (2002)
24. Schölkopf, B., Smola, A.J., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**(5), 1299–1319 (1998)
25. Sugiyama, M., Ghisu, M.E., Llinares-López, F., Borgwardt, K.: `graphkernels`: R and Python packages for graph comparison. Bioinformatics **34**(3), 530–532 (2017)
26. Turner, K., Mileyko, Y., Mukherjee, S., Harer, J.: Fréchet means for distributions of persistence diagrams. Discr. Comput. Geom. **52**(1), 44–70 (2014)
27. Yanardag, P., Vishwanathan, S.V.N.: Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1365–1374. ACM, New York, NY (2015)