Artificial Intelligence Review (2021) 54:1969–2009
https://doi.org/10.1007/s10462-020-09897-4

Persistence codebooks for topological data analysis

Bartosz Zieliński[1] · Michał Lipiński[1] · Mateusz Juda[1] · Matthias Zeppelzauer[2] · Paweł Dłotko[3]

Published online: 1 September 2020

Abstract
Persistent homology is a rigorous mathematical theory that provides a robust descriptor of data in the form of persistence diagrams (PDs) which are 2D multisets of points. Their variable size makes them, however, difficult to combine with typical machine learning workflows. In this paper we introduce persistence codebooks, a novel expressive and discriminative fixed-size vectorized representation of PDs that adapts to the inherent sparsity of persistence diagrams. To this end, we adapt bag-of-words, vectors of locally aggregated descriptors and Fischer vectors for the quantization of PDs. Persistence codebooks represent PDs in a convenient way for machine learning and statistical analysis and have a number of favorable practical and theoretical properties including 1-Wasserstein stability. We evaluate the presented representations on several heterogeneous datasets and show their (high) discriminative power. Our approach yields comparable—and partly even higher—performance in much less time than alternative approaches.

Keywords Persistent homology · Machine learning · Persistence diagrams · Bag of words · VLAD · Fisher vectors

1 Introduction

Topological data analysis (TDA) provides a powerful framework for the structural analysis of high-dimensional data. An important tool in TDA is persistent homology, PH (Edelsbrunner et al. 2002). It provides a comprehensive, multiscale summary of the underlying data's shape and currently gains an increasing importance in data science (Ferri 2017). Recently, it has been successfully applied to computer vision problems, such as shape and texture analysis (Li et al. 2014; Reininghaus et al. 2015), 3D surface

✉ Bartosz Zieliński
bartosz.zielinski@uj.edu.pl

[1] Institute of Computer Science and Computer Mathematics, Faculty of Mathematics and Computer Science, Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland

[2] Media Computing Group, Institute of Creative Media Technologies, St. Pölten University of Applied Sciences, Matthias Corvinus-Strasse 15, 3100 St. Pölten, Austria

[3] Dioscuri Centre in Topological Data Analysis, Institute of Mathematics, Polish Academy of Sciences, Jana i Jedrzeja Sniadeckich 8, 00-656 Warsaw, Poland

analysis (Adams et al. 2017; Zeppelzauer et al. 2017), 3D shape matching (Carrière et al. 2015), mesh segmentation (Skraba et al. 2010), and motion analysis (Vejdemo-Johansson et al. 2015). Further application areas include time series analysis (Seversky et al. 2016), music tagging (Liu et al. 2016) and social-network analysis (Hofer et al. 2017) as well as applications from the bio-medical domain, e.g. biomolecular analysis (Cang and Wei 2017), brain network analysis (Lee et al. 2012), protein investigation (Gameiro et al. 2015) and material science (Nakamura et al. 2015).

Persistent homology can be efficiently computed using various currently available tools (Bauer et al. 2017; Chen and Kerber 2011; De Silva et al. 2011; Dey et al. 2016; Edelsbrunner and Harer 2010; Maria et al. 2014). A basic introduction to PH is given in Sect. 2 and the more detailed one in the "Appendix". The common representation of PH are *persistence diagrams* (PDs) which are multisets of points in $\mathbb{R}^2$. Due to their variable size, which varies depending on the input data, PDs are not easy to integrate within common data analysis, statistics and machine learning workflows. To alleviate this problem, a number of kernel functions defined on PDs and vectorization methods for PDs have been introduced.

Kernel-based approaches have a strong theoretical background but in practice they often become inefficient when the number of training samples is large. As typically the entire kernel matrix must be computed explicitly (like in case of SVMs), this leads to roughly quadratic complexity in computation time and memory with respect to the size of the training set. Furthermore, vector-based approaches are limited to kernelized methods, such as SVM and kernel PCA. Vectorized representations, in contrast, are compatible with a much wider range of methods and do not suffer from complexity constraints of kernels. They, however, often lack in representational power, as they require the spatial quantization of the PDs, which is unsually non-adaptive and thus does not cope well with the sparseness of PDs.

In this work we present a novel adaptive representation of PDs which aims at combining the large representational power of kernel-based approaches with the general applicability of vectorized representations. To this end, we adapt the popular bag-of-words (BoW) encoding (McCallum et al. 1998; Sivic and Zisserman 2003), as well as its more comprehensive extensions, such as VLAD (Jégou et al. 2010) and Fisher vectors (Perronnin and Dance 2007) to cope with the inherent sparsity of PDs. The proposed persistent codebooks provide universally applicable fixed-sized feature vectors. They are, under mild assumptions, stable with respect to a standard metric in PDs and thus, also theoretically, built upon a solid basis. The presented method is to some extend a generalization of persistence images, PI (Adams et al. 2017), which adapts to the underlying data distribution. In contrast, PI samples the distribution in a regular grid (corresponding to an image), what often results in unnecessary codewords. Experiments show that the new representations achieve peak performance and even outperform numerous competitive methods while being more compact and requiring orders of magnitude less time.

This paper builds upon previous work of Zieliński et al. (2019). The additional contribution includes: (1) two new persistence codebook representations (PVLAD and PFV) building upon vectors of locally aggregated descriptors (VLAD) and Fisher vectors (FV); (2) the investigation of their stability; (3) the introduction of stable variant of PVLAD algorithm together with the proofs of its stability; (4) a significant number of additional experiments on an extended collection of datasets; and (5) an extended discussion of results.

The paper is structured as follows. Section 2 gives a basic introduction to PH and reviews related approaches. In Sect. 3 we introduce persistence codebooks and investigate their stability. Sections 4 and 5 present the experimental setup and results. We conclude the work in Sect. 6.

2 Background and related work

2.1 Background on persistent homology

In this section, we first introduce persistent homology, and then describe related state-of-the-art approaches, both kernel- and vectorization-based, that aim at making PH compatible with machine learning methods.

Under mild assumptions, persistent homology (PH) can be defined for a continuous function $f : X \to \mathbb{R}$, where $X \subset R^n$. Typically f is a distance function from a collection of points, or a scalar value function defined on a grid of points, but in principle it can be an arbitrary function that satisfies a tameness assumption specified below. Focusing on sublevel sets $L_x = f^{-1}((-\infty, x])$, we let x grow from $-\infty$ to $+\infty$. While this happens, we can observe a whole hierarchy of events. In dimension zero, connected components of L_x will be created and merged together. One dimensional cycles that are not bounded, or higher dimensional voids, will appear in L_x at critical points of f. The value of x on which a connected component, cycle or a higher dimensional void appears is refereed to as *birth time*. They will subsequently either become identical (up to a deformation) to other cycles and voids (created earlier), or they will be glued-in and become trivial. The value of x on which that happens is refereed to as *death time*. Every connected component, a cycle, or a higher dimensional void can, therefore, be characterized by a pair of numbers, b and d, its birth and death time. The difference between the death and the birth, $p = d - b$, is the so-called *persistence value*. In this paper, we will use the *birth-persistence* pair $[b, p]$ to encode the feature. The multi-set of birth-persistence pairs makes up a *persistence diagram* (PD). The set of all persistence diagrams will be denoted as $\mathcal{D}$. Example PDs for three different input point clouds are shown in Fig. 1.

The persistence coordinate is often an indicator of whether a cycle is structurally relevant or more likely to be related to noise. This observation is justified by many *stability theorems* for persistence (Cohen-Steiner et al. 2007), which state that a small change in the space X, or in a function f, implies only a small change in the resulting persistence diagram. Consequently, points in the PD with low persistence can be removed by a small perturbation of the data; and therefore, are not considered *stable features*. Those stability results make PDs a robust tool in data analysis.

Throughout this paper we assume that the given function f is *tame*, i.e. it induces a finite number of birth-persistence points. There are various metrics on finite PDs. To define them, the finite diagrams have to be enriched with an infinite collection of points $(b, 0)$, which represent features that are born and immediately die. Having the enriched PDs B and B' let us consider all possible matchings $\eta : B \to B'$. The 1-Wasserstein distance is defined as:

$$W_1(B, B') = inf_{\eta : B \to B'} \sum_{x \in B} \|(x - \eta(x))\|_\infty$$

In this paper, when considering stability of the representations, we will consider the stability with respect to 1-Wasserstein distance. A more in-depth introduction to PH is provided in “Appendix”.

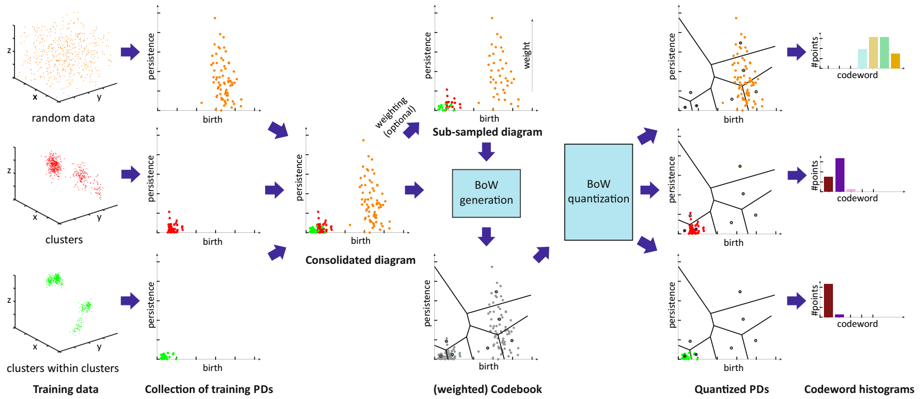

Fig. 1 The principle behind persistent codebooks on the example of the computational workflow of the persistence bag-of-words representation (PBoW): From the input data we compute PDs in dimension 1 in birth-persistence coordinates and combine them into one consolidated diagram (for the entire dataset). Next, a subset of points is obtained from this diagram by either a weighted or unweighted sub-sampling. Subsequently, we cluster the sub-sampled consolidated diagram to retrieve the codewords which will form our codebook. Finally, the points for each input PD are encoded by the codewords (BoW quantization). In this illustration a hard assignment of points to words (PBoW) is performed. The result is a codeword histogram for each input PD that represents how many points fall into which cluster of the codebook, i.e. codeword cardinalities. These codeword histograms are a compact and fixed-size vectorial representation. It is worth mentioning that while the hard assignment presented here gives the idea of the procedure, in practice we often employ soft assignment for stability reasons. Please, note further that the workflows for other persistent codebook encodings (e.g. based on VLAD or Fisher Vectors) are structurally similar, but partly use different codeword generation, quantization, and assignment schemes

2.2 Kernels and vectorized representations of PDs

Numerous kernel-based and vectorized approaches have been introduced to make PDs compatible with statistical analysis and machine learning methods. The goal of **kernel-based approaches** is to define dissimilarity measures (also known as kernel functions) on PDs to compare them, and thereby make them compatible with kernel-based machine learning methods, such as Support Vector Machines (SVMs), and kernel Principal Component Analysis (kPCA).

Li et al. (2014) combine the traditional bag-of-features (BoF) approach with PDs by using various distance functions between 0-dimensional PDs (bottleneck and Wasserstein distances for PDs, L^p distance functions for persistence landscapes of PDs) to generate kernels. On different datasets (SHREC 2010, TOSCA, hand gestures, Outex) they show that topological information is complementary to the information of traditional BoF. Reininghaus et al. (2015) propose a kernel for persistence diagrams by turning PDs into a continuous distribution by appropriate placement of Gaussian distributions in $\mathbb{R}^2$. Subsequently, they define a kernel as a scalar product of the two corresponding distributions. They apply topological descriptors together with the novel kernel to shape retrieval and texture classification. Kusano et al. (2016) propose a persistence weighted Gaussian kernel, which employs the framework of kernel embedding of measures into reproducing kernel Hilbert spaces. Carrière et al. (2017) propose another kernel based on sliced approximation of the Wasserstein distance. The authors show that the kernel is not only stable, but also mimics bottleneck distances between PDs. They subsequently develop an approximation technique to reduce the kernel computation time. They apply it to 3D shape segmentation, texture classification, and orbit recognition in dynamical systems. Another approach for the representation of PDs are persistence landscapes, PL (Bubenik 2015). PL is a stable functional representation of a PD obtained from transforming it into a sequence of real-valued piecewise linear functions. To compare two landscapes, the authors use standard L^p distance. This distance can be used to define a kernel function. Note that PL can also be transformed into a fixed-length vectorized representation by sampling the values of the landscape function. The authors, however, are not reporting results for vectorized PLs; therefore, we compare to kernels derived from PLs in our experiments. More recently, Le and Yamada (2018) proposed persistence Fisher kernel. It is based on a Fisher information distance between persistence diagrams and preserves some of the geometrical properties of the persistence diagrams space. Another approach, proposed by Som et al. (2018), embeds persistence diagrams into a Grassmann manifold, where PDs are compared using a geodesic distance.

Vectorized representations aim at deriving fixed-size encodings of PDs that can be used directly as input to current machine learning methods. One of the first attempt to vectorize PDs was presented by Aadcock et al. (2014). Given a collection of PDs $D_1, \dots, D_n$, a vector characterizing D_i is obtained by taking the vector of matching distances between D_i and D_j, for every $j \in \{1, \dots, n\}$. A more recent approach, called persistence image, PI (Adams et al. 2017) is built upon earlier work on size functions (Donatini et al. 1998; Ferri et al. 1998). It maps a PD to a space of functions from $\mathbb{R}^2$ to $\mathbb{R}$ by taking a weighted sum of two dimensional Gaussian kernels placed in the points of PD. Subsequently, a discretization of the obtained function on a fixed grid of points provides the vectorization of PDs. Anirudh et al. (2016) propose an alternative approach based on the reconstruction of a certain Riemannian manifold (RM) based on PDs and its subsequent representation by a fixed-size vector. In Di Fabio and Ferri (2015), PDs are represented as the coefficients of a complex polynomial having points of PD as roots. Similarly, using a sequence of weighting functions, Wang et al. (2019) transform PDs $D_1, \dots, D_n$ into vectors $V_1, \dots, V_n$ and obtain a polynomial representation of D_i by taking a polynomial system having roots in the corresponding V_i. Recently, as a continuation of this idea, the tropical algebra was used to construct a polynomial representation of diagrams (Kališnik 2019; Monod et al. 2019).

Recently, a third type of approach has been introduced, which aims at learning which points in the PD are of particular importance for the given task in a supervised manner by end-to-end learning (Hofer et al. 2017).

Overall, we can distinguish between approaches that learn the representation in a supervised or unsupervised manner. While supervised learned representations may better adapt to a specific task, the representations or kernels constructed in an unsupervised fashion bear less risk for overfitting because their construction is task agnostic. The proposed approach falls in the category of task agnostic representations and can be applied in supervised and unsupervised problem settings.

3 Persistence codebooks

In this section, we adapt the bag-of-words (BoW) model (McCallum et al. 1998; Sivic and Zisserman 2003) as well as its more comprehensive extensions, such as VLAD (Jégou et al. 2010) and Fisher vector (Perronnin and Dance 2007), introduced originally in text and image retrieval, for adaptive quantization of PDs into a fixed length vectorial representation. The idea behind BoW is to quantize variable length input data

Table 1 The design space of persistent codebook approaches introduced in this paper together with their abbreviations for reference

Sampling consolidated PD	Codebook generation	Codebook size	Histogram assignment	Abbrev.	Equation
No weighting weighting	k-means	See Tables 4 and 5	Hard	PBoW	(1)
			Hard with weights	wPBoW	(3)
			Hard	PVLAD	(5)
	GMM		Soft	sPBoW	(4)
				sPVLAD	(6)
				sFV	(11)

Each resulting representation can use either weighting of no weighting in codebook generation (see experiments in Sect. 5.1 for a direct comparison). For variants with weighting, we add "-w" to the abbreviation, e.g. "PBoW-w" for clarity

into a fixed-size representation by using a common dictionary, also called *codebook* of constant size. The codebook is generated from the input data in an unsupervised manner by extracting centers of clusters obtained from data clustering. The basic assumption behind BoW is that the clusters (i.e. codewords) capture the intrinsic structure of the data and, thereby represent an efficient vocabulary for the quantization of the data.

The overall approach of bag-of-words for persistence diagrams is visualized in Fig. 1. The input is a set of PDs extracted from all instances of a given dataset. First, all PDs are merged into one diagram. This consolidated diagram is then sub-sampled to reduce the influence of noise. In this paper, we consider two types of sub-sampling. A standard one which does not consider the persistence of the points, and one where points of higher persistence are more likely to be sampled, see Sect. 3.2 (we refer to those two types of sub-sampling as *without* and *with weighting*, respectively). From the (sub-sampled) consolidated diagram, the codebook C is generated using clustering. Given a codebook C, every input point P is encoded by assigning it to the nearest codeword from C. In traditional BoW this encoding leads to a codeword histogram, i.e. a histogram for which each codeword from C counts how many points from P are closest to this codeword. Further encodings investigated include vector of locally aggregated descriptors (VLAD) and Fisher vector (FV), see below.

For the proposed approaches, three important hyperparameters need to be identified: (1) the clustering algorithm used to generate the codebook, (2) the size of the codebook, i.e., the number of clusters, and (3) the type of proximity encoding which is used to obtain the final descriptors, i.e. hard and soft assignment. In this paper, we use k-means and Gaussian mixture models (GMM) for clustering. The codebook size is investigated empirically. In the following sections, we introduce persistent codebook approaches based on different quantizations and encodings, such as standard BoW, VLAD and FV. Consult Table 1 for an overview of representations introduced and evaluated in this paper.

For all approaches presented in these sections, we show if they are stable with respect to 1-Wasserstein distance. We would like to indicate that since the representations presented here are additive (consider the definition of additivity from Reininghaus et al. 2015), they are not stable for a p-Wasserstein distance for any $p > 1$ as indicated in Theorem 3 in Reininghaus et al. (2015).

3.1 Persistence bag of words (PBoW)

Let us first consider a direct adaptation of BoW (Baeza-Yates et al. 1999; Sivic and Zisserman 2003) to PDs. Given a collection of persistence diagrams $B_1, B_2, \dots, B_n$, they are consolidated into $D = B_1 \cup B_2 \cup \dots \cup B_n$ and a codebook of size N is obtained by using k-means clustering on D. Let $\{\mu_i \in \mathbb{R}^2, i = 1, \dots, N\}$ denote the centers of obtained clusters (the codewords). Moreover, for a PD $B = \{x_t \in \mathbb{R}^2\}_{t=1}^{T}$, let us denote $NN(x_t)$ as the index of the codeword nearest to x_t, $NN(x_t) = i|\ d(x_t, \mu_i) \le d(x_t, \mu_j)$ for all $j \in \{1, \dots, N\}$. For every codeword μ_i, $v_i^{PBoW}(B) = card\{x_t \in B \mid NN(x_t) = i\}$ captures the number of points from B, which are closer to μ_i than to any other μ_j. Then the *persistence bag of words* (PBoW) is defined as a vector:

$$\mathbf{v}^{\mathbf{PBoW}}(B) = \left(v_i^{PBoW}(B)\right)_{i=1,\dots,N}. \tag{1}$$

Subsequently, $\mathbf{v}^{\mathbf{PBoW}}(B)$ is normalized by taking the square root of each component (preserving the initial sign) and dividing it by the norm of the whole vector:

$$v_i^{PBoW}(B) = \frac{\operatorname{sign}\left(v_i^{PBoW}(B)\right)\sqrt{\left|v_i^{PBoW}(B)\right|}}{\left\|\mathbf{v}^{\mathbf{PBoW}}(B)\right\|}.$$

This is a standard normalization for BoW (Perronnin et al. 2010), which reduces the influence of outliers.

Remark Let $B, B' \in \mathcal{D}$ be persistence diagrams containing only finitely many off-diagonal points. The persistence bag of words, PBoW with N words is *not stable* with respect to 1-Wasserstein distance.

Proof Let us assume that we have two clusters with centers $\mu_1 = (0,0), \mu_2 = (1,0) \in \mathbb{R}^2$, and PD B containing only one point $x_1 = (\frac{1}{2} + \epsilon, 0)$, for some small $\epsilon > 0$. Then, $\mathbf{v}^{\mathbf{PBoW}}(B) = [0,1]$, because x_1 is closer to μ_2 than μ_1. However, a small perturbation in B, e.g. by -2ϵ, changes the assignment of x_1 from μ_2 to μ_1. In this case $B' = \{y_1 = (\frac{1}{2} - \epsilon, 0)\}$ and $\mathbf{v}^{\mathbf{PBoW}}(B') = [1,0]$. In order to be stable in 1-Wasserstein sense, PBoW should fulfill the following condition:

$$2 = |\mathbf{v}^{\mathbf{PBoW}}(B) - \mathbf{v}^{\mathbf{PBoW}}(B'))| < C|x_1 - y_1| < 2C\epsilon,$$

therefore $C > 1/\epsilon$. As $\epsilon > 0$ can be arbitrarily small, there does not exist a constant C that meets this condition. Hence, the direct adaption of BoW to PDs (PBoW) is not stable.

□

3.2 Weighted subsampling for codebook generation

Aside from being unstable, the straight-forward application of BoW to PD would neglect an important property of persistence diagrams, i.e. that points in a PD with higher persistence are typically considered more important than points with lower persistence. It is a consequence of a stability theorem, Edelsbrunner and Harer (2010) indicating that points with low persistence are more likely to originate from a noise than the points of high persistence.

In order to integrate this property into the codebook generation procedure, we perform k-means clustering on a subset of points obtained by a *weighted* sampling described below. This results in extended procedure of codebook generation which is as follows:

1. Place all the persistence diagrams (or all diagrams of a certain dimension) on to a single consolidated persistence diagram D.
2. Subsample n points from D in a way that points of higher persistence are more likely to be sampled. In the experiments presented in this paper we set it to $n = 10{,}000$.[1]
3. Perform k-means clustering on the obtained subset of n points to extract the centers of the clusters (the codewords).

For the weighted sampling of points from a persistence diagram we define a piecewise linear weighting function $w_{a,b} : \mathbb{R} \to \mathbb{R}$ as:

$$w_{a,b}(t) = \begin{cases} 0 & \text{if } t < a \\ (t-a)/(b-a) & \text{if } a \leq t < b \ , \\ 1 & \text{if} \, t \geq b \end{cases} \tag{2}$$

and use it to weight second coordinates (persistence) of points in PD. In our experiments we set a and b to the persistence values corresponding to 0.05 and 0.95 quantiles of the persistence coordinate of the points in D. In the performed sub-sampling persistence points having longer values of the function w are more likely to be sampled.

We want to highlight that in this case we have selected a *linear weighting* with respect to persistence, i.e. the probability of sampling of a point is *proportional to its persistence*. It works well in the cases considered in this paper, however in the case of very noisy data with just a few dominant persistent points, the points of high persistence may not be sampled at all. In such case, we suggest to consider the weighting to be a higher degree polynomial or an exponential function to boost the probability of capturing the high persistence points.

Please, note further that the sub-sampling does not directly enforce the points of the highest persistence to be automatically selected as the centers of clusters, but it makes the probability of such an event considerably larger. Examples of birth-persistence distributions with standard (unweighted) and weighted codebooks obtained with k-means and GMM are presented in Fig. 2. The unweighted clustering produces larger clusters, which are less adaptive to the strongest topological structures. At the same time, the weighted clustering yields a more adaptive codebook with a more uniform sampling of the space.

3.3 Weighted codeword assignment for persistence bag of words (wPBoW)

The weighting function from Sect. 3.2 can be similarly used to weight the histogram assignments to give points with higher persistence more influence in the final representation. For this purpose, instead of counting the number of points, we sum up the weights of their persistent coordinates.

[1] Preliminary experiments have shown that this number is insensitive and has little influence on the results (evaluated value range: 1000–100,000).

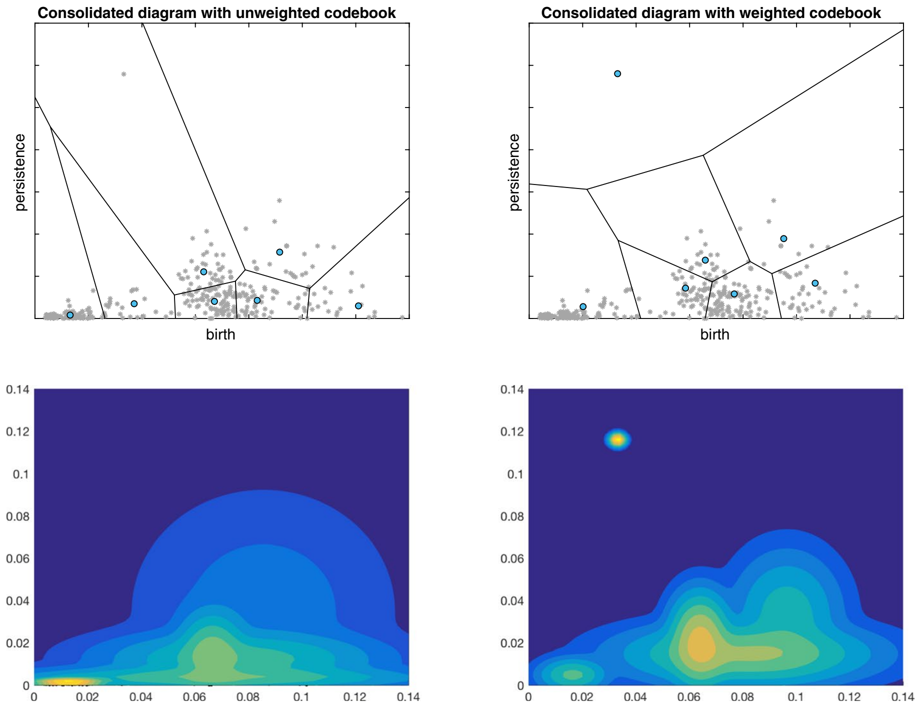

Fig. 2 Codebook generation based on the consolidated PD with $N = 7$ codewords (top: k-means, bottom: GMM, left: no weighting, right: weighting). Using the language of computational geometry; one may tell that the cells in the top diagrams form a Voronoi diagram of the codewords. Equivalently all points in the same Voronoi cell have the same closest codeword. Weighting allows to sample more points of higher persistent from the diagram, representing more stable topological structures and yields more balanced cluster weights (even though there are many more points on the bottom of consolidated PD). Note that a cluster can (but does not have to) be created for a single high persistence point which is well separated from the others, as is the case here with the most persistent point (top-left quadrant)

$$\mathbf{v}^{\mathbf{wPBoW}}(B) = \left(v_i^{wPBoW} = \sum_{(b,p)\in B\,:\,NN((b,p))=i} w_{a,b}(p) \right)_{i=1..N}, \tag{3}$$

where $B \in \mathcal{D}$. We will refer to this representation as *weighted persistence bag of words* (wPBoW) in the following. Similary to standard PBoW, wPBoW is not stable with respect to 1-Wasserstein distance. The couterexample is identical with the one in Sect. 3.1, when we assume that function $w_{a,b}$ is identity.

3.4 Stable persistence bag of words (sPBoW)

After having integrated persistence-based weighting into codebook generation and also into histogram assignment, we aim at making the representation stable. To this end we adapt soft assignment of points to clusters and prove that such an approach guarantees stability of the resulting representation. *Stable persistence bag of words* (sPBoW) similarly to PBoW (and wPBoW) first consolidates PDs in the initial step of construction, and then generates a GMM based on the sub-sampled points (e.g. by expectation maximization algorithm Nasrabadi 2007). This approach was originally introduced by Van Gemert et al. (2008).

Let the parameters of the fitted GMM be $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1, \dots, N\}$, where w_i, μ_i and Σ_i denote the weight, mean vector and covariance matrix of i-th Gaussian, and N denotes the number of Gaussians. Given a PD B, the stable PBoW is defined as:

$$\mathbf{v}^{\mathbf{sPBoW}}(B) = \left(v_i^{sPBoW} = w_i \sum_{x_t \in B} p_i(x_t|\lambda) \right)_{i=1,\dots,N}, \tag{4}$$

where $w_i > 0$, $\sum_{i=1}^{N} w_i = 1$, and $p_i(x_t|\lambda)$ is the likelihood that observation x_t was generated by Gaussian i:

$$p_i(x_t|\lambda) = \frac{exp\left\{ -\frac{1}{2}(x_t - \mu_i)' \Sigma_i^{-1} (x_t - \mu_i) \right\}}{2\pi |\Sigma_i|^{\frac{1}{2}}}.$$

The intuition behind this approach is to assign each point to *all* codewords, but with weight inversely proportional to the distance to the codewords.

Theorem *Let B and B' be persistence diagrams with a finite number of non-diagonal points. Stable persistence bag of words, sPBoW with N words is stable with respect to* 1*-Wasserstein distance between the diagrams, that is*

$$\left\| \mathbf{v}^{\mathbf{sPBoW}}(B) - \mathbf{v}^{\mathbf{sPBoW}}(B') \right\|_\infty \leq C \cdot W_1(B, B'),$$

where C is a constant.

Proof Let $\eta : B \to B'$ be the optimal matching in the definition of 1-Wasserstein distance. For a fixed $i \in \{1, \dots, N\}$ we have:

$$\begin{aligned} \left\| v_i^{sPBoW}(B) - v_i^{sPBoW}(B') \right\|_\infty &= \left\| w_i \sum_{x \in B} (p_i(x|\lambda) - p_i(\eta(x)|\lambda) \right\|_\infty \\ &\leq |w_i| \sum_{x \in B} \left\| \left(p_i(x|\lambda) - p_i(\eta(x)|\lambda) \right) \right\|_\infty \end{aligned}$$

As $p_i : \mathbb{R}^2 \to \mathbb{R}$ are Lipschitz continuous with the Lipschitz constants L_i, we get

$$\begin{aligned} |w_i| \sum_{x \in B} \left\| \left(p_i(x|\lambda) - p_i(\eta(x)|\lambda) \right) \right\|_\infty &\leq |w_i| \sum_{x \in B} \left\| \left(L_i(x - \eta(x)) \right) \right\|_\infty \\ &= |w_i \, L_i| \sum_{x \in B} \|(x - \eta(x))\|_\infty = |w_i \, L_i| \; W_1(B, B'). \end{aligned}$$

□

3.5 Persistence VLAD

Persistence VLAD (PVLAD) is based on vector of locally aggregated descriptors (VLAD) by Jégou et al. (2010), an extension of the bag-of-words concept, which accumulates the residual of each descriptor with respect to its assigned cluster. The first computation step is similar to PBoW: a codebook $\{\mu_i \in \mathbb{R}^2, i = 1..N\}$ is obtained from a training set using k-means clustering. Given a new PD B, each point $x_t \in B$ is associated with its nearest codeword $NN(x_t)$. In the second step, for each codeword μ_i, we compute a sum of differences between μ_i and all $x_t \in B$ for which $NN(x_t) = i$. This results in:

$$\mathbf{v^{PVLAD}} = \left(v_i^{PVLAD} = \sum_{x_t : NN(x_t)=i} x_t - \mu_i \right)_{i=1}^{N}. \tag{5}$$

The dimension of v_i^{PVLAD} equals 2 (differences on two coordinates), therefore $\mathbf{v^{PVLAD}}$ is of size $2N$. Intuitively, this vector should capture more information than PBoW alone, because it encodes the first order moments of the points assigned to a codeword instead of simply counting those points.

Similarly to PBoW, PVLAD is not stable with respect to 1-Wasserstein distance. Therefore, in Sect. 3.6, we propose to adapt a stable variant of VLAD, called soft VLAD.

Remark Let $B, B' \in \mathcal{D}$ be persistence diagrams containing only finite off-diagonal points. The persistence vector of locally aggregated descriptors, PVLAD with N words is not stable with respect to 1-Wasserstein distance.

Proof Starting from two clusters with centers $\mu_1 = (0,0), \mu_2 = (1,0) \in \mathbb{R}^2$ and a persistence diagram $B = \{(\frac{1}{2} + \epsilon, 0)\}$, for small $\epsilon > 0$, we get $v_1^{PVLAD} = [0,0]$ and $v_2^{PVLAD} = [\epsilon - \frac{1}{2}, 0]$. However, similarly to the case of PBoW, a small perturbation of B, e.g. by $[-2\epsilon, 0]$ will change B to $B' = \{(\frac{1}{2} - \epsilon, 0)\}$ and the corresponding components of PVLADs to $[\frac{1}{2} - \epsilon, 0]$ and [0, 0]. Calculating the difference between $\mathbf{v^{PVLAD}}(B)$ and $\mathbf{v^{PVLAD}}(B')$:

$$\begin{aligned} \left|\mathbf{v^{PVLAD}}(B) - \mathbf{v^{PVLAD}}(B')\right| &= \left|\left[[0,0], \left[\epsilon - \frac{1}{2}, 0\right]\right] - \left[\left[\frac{1}{2} - \epsilon, 0\right], [0,0]\right]\right| \\ &= \left|\left[\left[\frac{1}{2} - \epsilon, 0\right], \left[\epsilon - \frac{1}{2}, 0\right]\right]\right| = 1 \end{aligned}$$

In order to be stable in 1-Wasserstein sense, PVLAD should fulfill the following condition: $1 = |\mathbf{v^{PVLAD}}(B) - \mathbf{v^{PVLAD}}(B'))| < C|x_1 - y_1| < 2C\epsilon$, therefore $C > \frac{1}{2\epsilon}$. As $\epsilon > 0$ can be arbitrarily small, and there does not exist a constant C that meets this condition. Therefore, PVLAD is not stable. □

3.6 Stable persistence VLAD

Similarly to PBoW, the hard association with codewords can be replaced by soft association in VLAD (Jégou et al. 2012), to account for instability. To this end, we define *stable persistence VLAD* (sPVLAD) as follows:

$$\mathbf{v^{sPVLAD}}(B) = \left(v_i^{sPVLAD} = \sum_{x_t \in B} \gamma_i(x_t)(x_t - \mu_i) \right)_{i=1..N}, \tag{6}$$

where $\gamma_i(x_t)$ is the soft assignment of descriptor x_t to ith Gaussian:

$$\gamma_i(x_t) = p(i|x_t, \lambda) = \frac{w_i p_i(x_t|\lambda)}{\sum_{j=1}^{N} w_j p_j(x_t|\lambda)},$$

In the stability theorem for stable persistence VLAD (presented below) we assume that coordinates of the points in the considered persistence diagrams are limited to a certain compact subset of $\mathbb{R}^2$. This limitation is crucial to prove the stability and it is a reasonable assumption in case of TDA. Moreover, this theorem is true for any $\mathbb{R}^n$ (not only for $\mathbb{R}^2$).

Theorem *Let $B, B' \in \mathcal{D}$ be persistence diagrams, such that $B, B' \subset [a,b] \times [a,b]$ and let $\lambda = \{w_i, \mu_i, \Sigma_i, i = 1, \dots, N\}$ be a GMM with $w_i \neq 0$. The stable persistence VLAD with N words is stable with respect to* 1*-Wasserstein distance, that is:*

$$\left\| \mathbf{v}^{\mathbf{sPVLAD}}(B) - \mathbf{v}^{\mathbf{sPVLAD}}(B') \right\|_\infty \le C \cdot W_1(B, B'),$$

where C is a constant depending on $[a,b] \times [a,b]$.

Proof Let us first consider the following difference:

$$\begin{aligned}
|\gamma_i(x_t) - \gamma_i(y_t)| &= \left| \frac{w_i p_i(x_t)}{\sum_{j=1}^{N} w_j p_j(x_t)} - \frac{w_i p_i(y_t)}{\sum_{j=1}^{N} w_j p_j(y_t)} \right| \\
&= \left| \frac{w_i p_i(x_t) \sum_{j=1}^{N} w_j p_j(y_t) - w_i p_i(y_t) \sum_{j=1}^{N} w_j p_j(x_t)}{\sum_{j=1}^{N} w_j p_j(x_t) \sum_{j=1}^{N} w_j p_j(y_t)} \right| \\
&= \left| \frac{w_i \sum_{j=1}^{N} \big(w_j p_i(x_t) p_j(y_t) - w_j p_i(y_t) p_j(x_t)\big)}{\sum_{j=1}^{N} \sum_{k=1}^{N} w_j w_k p_j(x_t) p_k(y_t)} \right| \\
&\le C_1 \left| \sum_{j=1}^{N} \big(w_j p_i(x_t) p_j(y_t) - w_j p_i(y_t) p_j(x_t)\big) \right| \\
&\overset{(i)}{=} C_1 \left| \sum_{j=1}^{N} \big(w_j p_i(x_t)\big(p_j(y_t) - p_j(x_t) + p_j(x_t)\big) - w_j p_i(y_t) p_j(x_t)\big) \right| \\
&= C_1 \left| \sum_{j=1}^{N} w_j \big(p_i(x_t)\big(p_j(y_t) - p_j(x_t)\big) + p_j(x_t)\big(p_i(x_t) - p_i(y_t)\big)\big) \right| \\
&\le C_1 \left| \sum_{j=1}^{N} w_j \big(M_i L_j \|y_t - x_t\|_\infty + M_j L_i \|x_t - y_t\|_\infty\big) \right| \\
&\le C_1 N \max_j \{w_j \big(M_i L_j + M_j L_i\big)\} \|y_t - x_t\|_\infty = C_2 \|y_t - x_t\|_\infty,
\end{aligned} \tag{7}$$

where:

$$\begin{aligned}
C_1 &= \frac{\max_i\{w_i\}}{N^2 \min_j\{w_j^2\} \min_j\{\min\{p_j^2(z) \,|\, z \in [a,b] \times [a,b]\}\}}, \\
C_2 &= C_1 N \max_j \{w_j \big(M_i L_j + M_j L_i\big)\},
\end{aligned}$$

while M_i and L_i are the maximal value and Lipschitz constant of i-th Gaussian p_i. Note that the constant C_1 exists because diagrams are supported in a compact subset of $\mathbb{R}^2$. Therefore, the Gaussians achieve a minimum value, which is bounded away from zero. When it comes to assignment (i), we simply put $p_j(y_t) = p_j(y_t) - p_j(x_t) + p_j(x_t)$.

For a fixed i we can estimate:

$$\begin{aligned}
\left\|v_i^{sPVLAD}(B) - v_i^{sPVLAD}(B')\right\|_\infty &= \left\|\sum_t \gamma_i(x_t)(x_t - \mu_i) - \sum_t \gamma_i(y_t)(y_t - \mu_i)\right\|_\infty \\
&= \left\|\sum_t \gamma_i(x_t)(x_t - y_t + y_t - \mu_i) - \sum_t \gamma_i(y_t)(y_t - \mu_i)\right\|_\infty \\
&= \left\|\sum_t \big(\gamma_i(x_t) - \gamma_i(y_t)\big)(y_t - \mu_i) + \sum_t \gamma_i(x_t)(x_t - y_t)\right\|_\infty \\
&\overset{(ii)}{\leq} \left\|\sum_t C_2(y_t - \mu_i)\|y_t - x_t\|_\infty + \sum_t (x_t - y_t)\right\|_\infty \\
&\overset{(iii)}{\leq} \sum_t (C_3 + 1)\|y_t - x_t\|_\infty = C_4 \sum_t \|y_t - x_t\|_\infty \leq C_4 W_1(B, B'),
\end{aligned} \tag{8}$$

where:

$$\begin{aligned}
C_3 &= C_2 \max_i\{\max\{\|p - \mu_i\|_\infty \mid p \in [a,b] \times [a,b]\}\}, \\
C_4 &= C_3 + 1.
\end{aligned}$$

In (ii) we used the estimation (7) and the fact that $\sup_y \gamma_i(y) = 1$. The boundaries of $\mathbb{R}^2$ compact subset allow to determine the maximal distance between diagram points and the Gaussian centers, which is used in (iii) to estimate C_3. Note that C_4 is independent of i, hence:

$$\begin{aligned}
\left\|\mathbf{v}^{\mathbf{sPVLAD}}(B) - \mathbf{v}^{\mathbf{sPVLAD}}(B')\right\|_\infty &= \max_i \left\{ \left\|v_i^{sPVLAD}(B) - v_i^{sPVLAD}(B')\right\|_\infty \right\} \\
&\leq C_4 \cdot W_1(B, B'),
\end{aligned}$$

and

$$\begin{aligned}
\left\|\mathbf{v}^{\mathbf{sPVLAD}}(B) - \mathbf{v}^{\mathbf{sPVLAD}}(B')\right\|_p &= \sqrt[p]{\sum_i (\left\|v_i^{sPVLAD}(B) - v_i^{sPVLAD}(B')\right\|_\infty)^p} \\
&\leq \sqrt[p]{N\, C_4} \cdot W_1(B, B').
\end{aligned}$$

□

We would like to point out that in the estimation of C_1 and (iii) we use an assumption that diagrams are supported in the compact $\mathbb{R}^2$ subset $[a,b] \times [a,b]$. As a result, if the support of a persistence diagram sequence diverges to ∞, then the corresponding sequence of C_1 also diverges to infinity. Therefore C_4 is not a *global* constant and the persistence VLAD is not *globally* stable. We want to indicate, however, that for any practical case, the assumption about compact support of diagrams is always satisfied.

3.7 Persistence fisher vector

The idea of *persistence Fisher vector* (PFV) is based on Fisher vectors introduced by Perronnin and Dance (2007) and relies on the gradient of the log-likelihood with respect to the parameters of a Gaussian mixture model. Compared to the traditional BoW model, it captures first and second order moments. It can be extended to PDs as follows. Given a PD $B \in \mathcal{D}$ we aim a characterizing it with a gradient vector derived from a generative probability model (obtained for all PDs used for codebook generation). This is similar to sPVLAD, however in case of PFV we compute not only the first order, but also the second order moments of the points assigned to a codeword (i.e. not only the gradient for μ_i but also for Σ_i).

Let $\mathcal{L}(B|\lambda) = log\, p(B|\lambda)$, where under the independence assumption:

$$\mathcal{L}(B|\lambda) = log\ \Pi_{x_t \in B} p(x_t|\lambda)) = \sum_{x_t \in B} log\, p(x_t|\lambda),$$

where:

$$p(x_t|\lambda) = \sum_{i=1}^{N} w_i p_i(x_t|\lambda),$$

is the likelihood that point x_t was generated by the GMM.

Assuming that the covariance matrices are diagonal (for ease of calculation), the derivations $\frac{\partial \mathcal{L}(B|\lambda)}{\partial \mu_i^d}$ and $\frac{\partial \mathcal{L}(B|\lambda)}{\partial \sigma_i^d}$ (where $\sigma_i^d = diag(\Sigma_i)$ and superscript d denotes the d-th dimension of a vector) can be effectively computed as (Perronnin and Dance 2007):

$$\frac{\partial \mathcal{L}(B|\lambda)}{\partial \mu_i^d} = \sum_{x_t \in B} \gamma_i(x_t) \left[\frac{x_t^d - \mu_i^d}{(\sigma_i^d)^2} \right], \tag{9}$$

$$\frac{\partial \mathcal{L}(B|\lambda)}{\partial \sigma_i^d} = \sum_{x_t \in B} \gamma_i(x_t) \left[\frac{(x_t^d - \mu_i^d)^2}{(\sigma_i^d)^3} - \frac{1}{\sigma_i^d} \right]. \tag{10}$$

The gradient vector is just a concatenation of the partial derivatives with respect to all the parameters.

To normalize the dynamic range of the different dimensions of the gradient vectors, the diagonal of the Fisher information matrix F_λ is computed as:

$$F_\lambda = E_B[\nabla_\lambda \mathcal{L}(B|\lambda) \nabla_\lambda \mathcal{L}(B|\lambda)'].$$

applied to partial derivatives, resulting in the final definition of Fisher vector:

$$\mathbf{v}^{\mathbf{PFV}} = \left(f_{\mu_i^d}^{-1/2} \frac{\partial \mathcal{L}(B|\lambda)}{\partial \mu_i^2}, f_{\sigma_i^d}^{-1/2} \frac{\partial \mathcal{L}(B|\lambda)}{\partial \sigma_i^d} \right)_{i=1..N}, \tag{11}$$

where $f_{\mu_i^d}$ and $f_{\sigma_i^d}$ are the corresponding terms on the diagonal of F_λ. Vector $\mathbf{v}^{\mathbf{PFV}}$ is the concatenation of N pairs of components containing $D = 2$ values for every Gaussian component, therefore it is of size $4N$.

Theorem *Let $B, B' \in \mathcal{D}$ be persistence diagrams, such that $B, B' \subset [a,b] \times [a,b]$. The persistence Fisher vector with N words is stable with respect to 1-Wasserstein distance, that is:*

$$\left\| FV(B) - FV(B') \right\|_{\infty} \leq C \cdot W_1(B, B'),$$

where C is a constant depending on $[a,b] \times [a,b]$.

Proof Persistence Fisher Vector is a concatenation of the two components presented in Eqs. (9) and (10). In order to be stable, both components have to be stable with respect to 1-Wasserstein distance; therefore, we estimate them separately (we skip d superscript from the original notation for clarity).

The first FV component (10) can be estimated using the theorem about sPVLAD stability (8):

$$\begin{aligned}
&\left\| \sum_t \gamma_i(x_t) \left[\frac{x_t - \mu_i}{(\sigma_i)^2} \right] - \sum_t \gamma_i(y_t) \left[\frac{y_t - \mu_i}{(\sigma_i)^2} \right] \right\| \\
&\quad = \frac{1}{(\sigma_i)^2} \left\| \sum_t \gamma_i(x_t)(x_t - \mu_i) - \sum_t \gamma_i(y_t)(y_t - \mu_i) \right\| \\
&\quad = \frac{1}{(\sigma_i)^2} \left\| v_i^{sPVLAD}(B) - v_i^{sPVLAD}(B') \right\|_{\infty} \overset{(8)}{\leq} \frac{C_4}{(\sigma_i)^2} W_1(B, B'),
\end{aligned}$$

The second component (10) can be estimated as follows:

$$\begin{aligned}
&\left\| \sum_t \gamma_i(x_t) \left[\frac{(x_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i} \right] - \sum_t \gamma_i(y_t) \left[\frac{(y_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i} \right] \right\| \\
&= \left\| \sum_t \gamma_i(x_t) \left[\frac{(x_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i} \right] - \sum_t \gamma_i(y_t) \left[\frac{(y_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i} - \frac{(x_t - \mu_i)^2}{(\sigma_i)^3} + \frac{1}{\sigma_i} + \frac{(x_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i} \right] \right\| \\
&= \left\| \sum_t (\gamma_i(x_t) - \gamma_i(y_t)) \left[\frac{(x_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i} \right] - \sum_t \gamma_i(y_t) \left[\frac{(y_t - \mu_i)^2}{(\sigma_i)^3} - \frac{(x_t - \mu_i)^2}{(\sigma_i)^3} \right] \right\| \\
&\overset{(iv)}{\leq} \left\| \sum_t D_1 L_i(x_t - y_t) - \sum_t \frac{\gamma_i(y_t)}{(\sigma_i)^3} \left[(y_t - \mu_i)^2 - (x_t - \mu_i)^2 \right] \right\| \\
&= \left\| \sum_t D_1 L_i(x_t - y_t) - \sum_t \frac{\gamma_i(y_t)}{(\sigma_i)^3} \left[y_t^2 - x_t^2 - 2\,\mu_i(y_t - x_t) \right] \right\| \\
&= \left\| \sum_t D_1 L_i(x_t - y_t) - \sum_t \frac{\gamma_i(y_t)}{(\sigma_i)^3} \left[(y_t - x_t)(x_t + y_t) - 2\,\mu_i(y_t - x_t) \right] \right\| \\
&= \left\| \sum_t D_1 L_i(x_t - y_t) - \sum_t \frac{\gamma_i(y_t)}{(\sigma_i)^3} \, (y_t - x_t) \left[(x_t + y_t) - 2\,\mu_i \right] \right\| \\
&\overset{(v)}{\leq} \left\| \sum_t D_1 L_i(x_t - y_t) - \sum_t D_2\,(y_t - x_t) \right\| = \left\| \sum_t D_1 L_i(x_t - y_t) + \sum_t D_2\,(x_t - y_t) \right\| \\
&\leq D_3 \left\| \sum_t (x_t - y_t) \right\| \leq D_3 W_1(B, B'),
\end{aligned}$$

where D_1 is an upper bound for $\frac{(x_t - \mu_i)^2}{(\sigma_i)^3} - \frac{1}{\sigma_i}$ (iv), D_2 is a bound for $\frac{\gamma_i(y_t)}{(\sigma_i)^3}$ (v), both on the domain $[a,b] \times [a,b]$, and $D_3 = D_1\ max\{L_i\} + D_2$. □

Summing up the two estimates above, we conclude that persistence Fisher vector is stable with respect to 1-Wasserstein distance with a constant $D_3 + \frac{C_4}{(\sigma_i)^2}$, where D_3 is defined above and C_4 is defined in Sect. 3.6.

4 Experimental setup

To evaluate the proposed persistence BoW representations (PBoW, sPBoW, wPBoW, PVLAD, sPVLAD and PFV), we compare them with a number of state-of-the-art approaches including kernel-based methods and vectorized PD representations. The evaluation is performed on classification tasks involving different datasets representing heterogeneous data including, among others, 3D shapes, textures, and social media graphs. In the following, we describe the datasets used in our experiments, list the state-of-the-art approaches we compare with, and discuss the setup of the experiments.

4.1 Datasets

For the evaluation we incorporate various datasets which cover a wide range of different data types. Firstly, to provide a proof-of-concept, we evaluate all the approaches on a synthetically generated shape classes from Adams et al. (2017). Next, the approaches are evaluated on real-world datasets for 3D shape segmentation (Carrière et al. 2017), activity recognition in 3D motion capture data (Ali et al. 2007), geometry-informed material recognition (DeGol et al. 2016), classification of social network graphs (Hofer et al. 2017) and analysis of 3D surface texture (Zeppelzauer et al. 2017). The datasets are described in detail in the following sections. Where possible, we have used pre-computed PDs available with the datasets to foster reproducibility and comparability. As the computation times for some of the considered methods, especially for kernel-based approaches, do not scale well with the sizes of datasets, we have decided to randomly sub-sample some of the datasets (see details below).

4.1.1 Synthetic dataset

The first dataset is a synthetic dataset introduced by Adams et al. (2017). It consists of seven shape classes represented by point clouds in $\mathbb{R}^3$ of the following geometrical objects: unit cube, circle of diameter one, sphere of diameter one, three clusters with centers randomly chosen from unit cube, hierarchical structure of three minor clusters within three major clusters (where the centers of the minor clusters are chosen as small perturbations from the major cluster centers), and a torus (see Fig. 3 for example shapes). Each point cloud is randomly perturbed by positioning a Gaussian distribution of standard deviation 0.1 at this point and sampling novel points from the distribution. Overall, this dataset contains 50 point clouds for each of the six classes, each containing 500 3D points. This gives 300 point clouds in total.

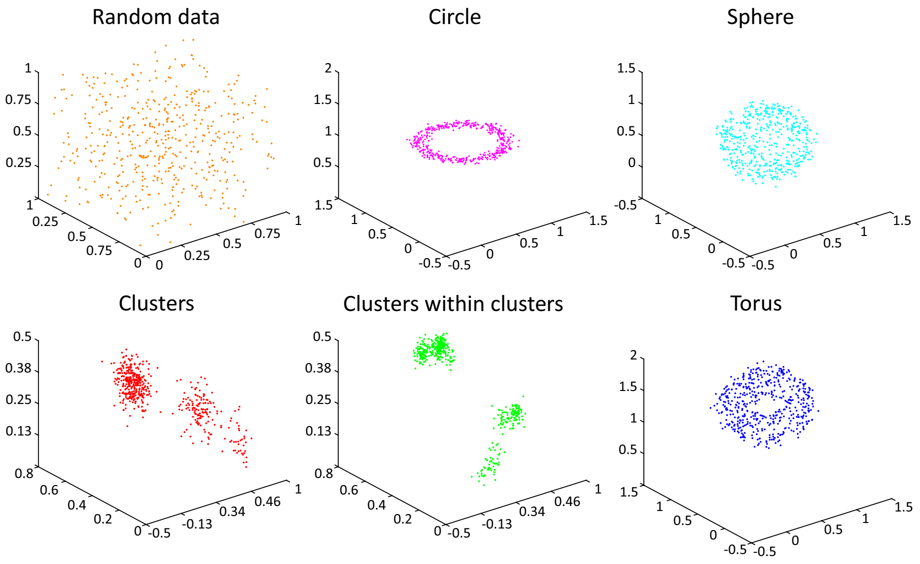

Fig. 3 Example shapes from the six shape classes of the synthetic dataset

From each point cloud, we compute the PDs in dimension 1 for a Vietoris-Rips filtration for a radius parameter equal to the maximal distance between points in the point cloud.[2] We employ the approximation method proposed by Dey et al. (2016) and the SimBa implementation based on the work of Dayu Shi.[3]

4.1.2 Geometry-informed material recognition dataset (GeoMat)

The GeoMat dataset provides geometry information (point clouds) as well as visual images of 19 different materials, such as “brick”, “grass” and “gravel” (DeGol et al. 2016). The GeoMat dataset contains patches sampled from larger photographs of surfaces from buildings and grounds. Each patch predominantly represents only one material, while each class consists of 600 images, each of size 100 × 100 pixels. Among them, there are pictures of different scales, i.e. 100 × 100, 200 × 200, 400 × 400 and 800 × 800.

For each patch, the dataset provides a depth image,[4] containing the local (fine-grained) surface texture and the global surface curvature. To filter out the global curvature, we transform each depth image into a point cloud in 3D space, consisting of 10, 000 points (every point represents one of the 100 × 100 pixels). Then, the resulting point cloud is rotated in a way that the Z axis represents depth and the global surface curvature is removed by fitting a second degree function (paraboloid) to the point cloud and subtracting the approximated values from the Z coordinates of the original points. The values of the Z-coordinates are

[2] We use Vietoris-Rips complexes to be consistent with the previous works. However, it should be noted that the alpha complex is a more optimal choice to compute persistent homology for a collection of points in low dimensional Euclidean space.

[3] http://web.cse.ohio-state.edu/~dey.8/SimBa/Simba.html, last visited September, 2019.

[4] Source: http://web.engr.illinois.edu/~degol2/pages/MatRec_CVPR16.html, last visited September, 2019.

then centered at 0 and the point cloud is projected back into a bitmap (depth map). Ultimately, PDs are computed by gray-scale filtration.

4.1.3 Social network graph datasets (Reddit)

To extend the range of different data types in our evaluation, we further incorporate graph-based datasets. To this end we employ the reddit-5k and reddit-12k datasets from Yanardag and Vishwanathan (2015), which contain discussion graphs from the reddit platform.[5] Nodes in the graphs correspond to users, and edges between users exist if one user has commented a posting of the other user. Different graphs are labeled by subreddits, which refer to different topics. The dataset reddit-5k contains overall 4999 graphs for 5 popular subreddits. The larger dataset, reddit-12k, contains 11929 graphs for 11 subreddits including topics like, e.g. "worldnews", "videos" and "atheism". The task for both datasets is to predict the subreddit (topic) from the input graph. For both datasets we use the pre-computed PDs available online.[6] They are obtained using filtration based on the vertex degree, that is, the number of edges incident to a vertex. More precisely, concerning vertices V, edges E, and vertex $v \in V$, $\deg(v)$ denote the number of edges in E that contain v. The filtration $f(v) = \deg(v)$ on vertices can be then extended to edges.That yields nontrivial persistent homology in dimension zero and one, where all one-dimensional classes are essential.

4.1.4 3D surface texture dataset (PetroSurf3D)

A further dataset in our experiments is the recently released *PetroSurf3D* dataset, which contains high-resolution 3D surface reconstructions from the archaeological domain with a resolution of approximately 0.1 mm (Poier et al. 2017). The reconstructions represent 26 natural rock surfaces that exhibit human-made engravings (so-called rock art), and thereby exhibit complex 3D surface textures. The classification task for PetroSurf3D is to automatically predict which areas of the surface have been manipulated by tools (engravings) and which have not, i.e. there are two classes of surface topographies: engraved areas and the natural rock surface. Engraved areas represent approximately 19% of the data. For each surface, a precise pixel-accurate ground truth exists together with a depth map of the surface. The depth maps are analyzed in a patch-wise manner. Overall, there are 754.386 square patches to classify from 26 surfaces. In order to keep the number of training samples in a practical range, we randomly subsampled each surface. Overall, a balanced set (equal class cardinalities) of 600 patches per surface ($26 * 600 = 15{,}600$ samples) is used in each repetition of an experiment.

For each patch, a PD is computed by grayscale filtration over the surface depth ranges (depth maps) as a basis for our experiments. To normalize the values for different shaped surfaces, the depth value range is z-standardized before filtration.

[5] Reddit is a content-aggregation website: http://reddit.com.

[6] Source: https://github.com/c-hofer/nips2017, last visited September, 2019.

4.1.5 3D shape segmentation dataset

We further employ the 3D shape dataset from Chen et al. (2009) which was preprocessed by Carrière et al. (2015) for topological data analysis. The preprocessed dataset contains PDs for 5700 3D points from airplane models. Each point is assigned to one sub-part (segment) of an airplane, e.g., 'wing', 'vertical stabilizer' and 'horizontal stabilizer'. For our experiments we use the PDs computed by Carrière et al. (2015), available in their repository.[7] The PDs were generated by tracking topology evolution of a geodesic ball centered at the individual points of the input 3D model. Thereby, the radius grows from 0 to infinity. We focus on PDs of dimension 1 as the considered 3D shapes are connected. The task is to classify each point according to the segment it belongs to.

4.1.6 Motion capture dataset

Another real-world dataset represents 3-dimensional motion capture sequences of body joints (Ali et al. 2007). The dataset describes the following five activities: dancing, jumping, running, sitting and walking with 31, 14, 30, 35 and 48 instances, respectively. For each activity, a set of 19 3D motion trajectories (each corresponding to the motion of one tracked joint) is extracted. This corresponds to $3 \cdot 19 = 57$ curves of individual (x, y, and z) components for which 57 separate PDs are computed by Ali et al. (2007). For our experiments we employ the original pre-computed PDs.[8]

Experiments with this dataset are performed only on the vectorized representations. For kernel-based approaches we would have to compute 57 full kernel matrices, which is computationally expensive and would further require an adequate method for the combination of the kernels. For vectorized representations, the proceedure is much more efficient and straight-forward. We simply compute one vector per PD and concatenate them into a final feature vector for classification. For the BoW approaches, we compute 57 codebooks, one for each 3D motion component, and concatenate the corresponding codeword histograms. In case of the Riemannian manifold representation, RM Anirudh et al. (2016), we generate vectorial features by PCA and concatenate them as proposed by the authors. We also use original procedure for PI (Adams et al. 2017).

4.2 Compared approaches

We compare our bag-of-word approaches with both kernel-based techniques and vectorized representations. Kernel-based approaches include: 2-Wasserstein distance,[9] 2Wd (Kerber et al. 2017.), the multi-scale kernel,[10] MK (Reininghaus et al. 2015), and sliced Wasserstein kernel,[11] SWK (Carrière et al. 2017). Furthermore, we employ the persistence landscape[12,13] (PL) representation and generate a kernel matrix by the distance metric

[7] Source: https://github.com/MathieuCarriere/sklearn_tda, last visited September, 2019.

[8] Source: https://github.com/rushilanirudh/pdsphere, last visited September, 2019.

[9] Source: https://bitbucket.org/grey_narn/hera, last visited September, 2019

[10] Source: https://github.com/rkwitt/persistence-learning, last visited September, 2019.

[11] Code obtained from Mathieu Carrière.

[12] Source: https://www.math.upenn.edu/~dlotko/persistenceLandscape.html, last visited September, 2019.

[13] Source: https://github.com/queenBNE/Persistent-Landscape-Wrapper, last visited September, 2019.

defined in Bubenik (2015). Vectorized PD representations include: persistence image[14], PI Adams et al. (2017) and the Riemannian manifold approach,[15] RM Anirudh et al. (2016). The original PI implementation is rather inefficient since it takes into account the exact location of all birth-persistence points when calculating the values of PI. Therefore, we additionally perform experiments with an approximated unstable version of PI (referred to as approxPI in the following), which applies the Gaussian filter to 2D histogram of birth-persistence points.[16]

We refrained from incorporating descriptors composed of simple topological statistics (such as minimum, maximum, and average of birth, death, and persistence) because, according to our previous research (Zeppelzauer et al. 2017), they lack sensitivity and are easily outperformed by more sophisticated approaches like persistence images.

4.3 Setup

For all datasets, except Reddit, in Sect. 4.1 we consider the PDs of dimension 1 as a common input (cycles) since they best express the internal structure in the data and yielded the most promising results in related works (Adams et al. 2017; Carrière et al. 2015). In case of Reddit database we use PDs of dimension 0 (connected components), since graphs are considered as 1-complex; thus, first dimensional homology generators never die. In the considered datasets no infinite intervals of dimension 1 occur. In cases where infinite intervals are present, there are different ways to proceed: (1) ignoring them, (2) substituting infinity with some (large) number or (3) building separate representations for finite and infinite intervals. In the general case, we recommend to compute persistence codebooks for PDs of all available dimensions separately and to combine them before classification.

The classification pipeline is as follows. For the kernel-based approaches, we take the PDs as input and compute the explicit kernel matrices for the training and test samples. Next, we train an SVM from the explicitly computed kernel-matrix and evaluate it on the test set. For the vectorized representations we compute the respective feature vectors from the PDs and feed them into a linear SVM for training. This procedure allows direct comparison between kernel-based approaches and vectorized representations.

For all datasets, we aim at solving a supervised classification task. In order to enhance the comparability with results obtained in original experiments, if available, we employ train/test divisions of samples based on the original procedures. To find optimal parameters for each evaluated approach, we run a grid search over their respective hyperparameters. The hyperparameters and their evaluated values for each approach are listed in Table 4 (for the kernel-based approaches) and in Table 5 (for the vectorized representations). The optimal parameters are highlighted in bold. For each parameter combination, we run a complete experiment including cross-validation on the training set to evaluate its performance. The number of repetitions for each parameter combination in the grid search depends on the dataset and is provided in Tables 2 and 3.

Our evaluation is partitioned into two sets of experiments. EXP-A uses all related approaches on a sub-sampled version of the datasets, while EXP-B operates only on the vectorized representations and uses larger datasets. The reason for this is that for the larger

[14] Source: https://github.com/CSU-TDA/PersistenceImages, last visited September, 2019.

[15] Source: https://github.com/rushilanirudh/pdsphere, last visited September, 2019.

[16] Code available at: https://github.com/bziiuj/pcodebooks.

Table 2 Results of EXP-A averaged over 5 runs

DESCR. NAME	Synthetic data (5 REPS.)		GeoMat (SUBSET 30 SMPL./CL.) (5 REPS.)		Reddit-5k (SUBSET 100 SMPL./CL.) (5 REPS.)		Reddit-12k (SUBSET 50 SMPL./CL.) (5 REPS.)		PetroSurf3D (SUBSET 390 SMPL./CL.) (4 REPS.)		3D Shape Segm. (5 REPS.)	
	Score	Time (s)	Score	Time (s)	Score	Time (s)	Score	Time (s)	Score	Time (s)		
2Wd	**98.00 ± 1.28**	133.40	**24.74 ± 3.14**	14,740.60	29.60 ± 5.55	6198.80	23.27 ± 2.37	4978.80	**79.88 ± 6.58**	63821.50	71.01 ± 0.45	6300.80 s
MK	92.34 ± 2.82	44.04	9.12 ± 3.08	10,883.00	32.67 ± 3.06	7251.67	26.67 ± 4.58	4222.33	**80.16 ± 3.57**	53,831.00	88.48 ± 0.47	860.33 s
SWK	**97.43 ± 1.86**	33.31	22.00 ± 2.72	1069.61	39.20 ± 6.10	373.77	30.91 ± 4.98	329.68	**77.98 ± 6.00**	3913.04	**94.22 ± 0.48**	1995.42 s
PL	95.14 ± 2.17	81.01	17.89 ± 3.83	1563.67	30.40 ± 6.23	790.34	24.73 ± 4.19	717.98	**78.21 ± 6.67**	7990.66	92.55 ± 0.79	2668.32 s
PI	**98.29 ± 1.56**	10.06	11.05 ± 2.44	342.27	**48.40 ± 5.73**	1250.88	**35.64 ± 4.56**	111.78	**80.00 ± 3.67**	1876.98	**94.86 ± 0.29**	291.22 s
approxPI	**98.57 ± 1.01**	0.79	16.00 ± 3.81	5.07	**44.80 ± 8.07**	6.87	**34.55 ± 6.68**	1.33	73.57 ± 4.93	5.59	92.45 ± 0.52	182.97 s
RM	92.29 ± 2.60	0.88	19.05 ± 2.72	10.34	39.20 ± 9.23	7.76	28.73 ± 1.99	17.94	**77.26 ± 6.29**	128.83	72.29 ± 0.42	6.57 s
PBoW	**98.00 ± 2.17**	0.82	**26.74 ± 2.51**	0.27	**46.80 ± 4.82**	**0.53**	**32.73 ± 2.23**	**0.32**	**79.88 ± 4.64**	**1.78**	90.78 ± 0.70	5.37 s
wPBoW	**97.71 ± 0.78**	**0.20**	**26.42 ± 2.45**	1.96	**46.80 ± 4.38**	1.22	**32.00 ± 4.38**	1.33	**79.64 ± 6.33**	3.54	90.09 ± 0.96	4.25 s
sPBoW	**96.86 ± 1.56**	3.96	22.21 ± 2.59	2.18	**45.60 ± 5.37**	0.74	**31.64 ± 2.76**	1.29	**78.81 ± 3.68**	31.50	**94.42 ± 0.56**	14.74 s
PVLAD	94.00 ± 1.56	0.49	21.05 ± 5.59	**0.20**	38.40 ± 8.65	1.23	25.45 ± 2.87	1.18	**78.21 ± 3.84**	3.94	87.35 ± 0.43	**3.57 s**
sPVLAD	95.71 ± 2.47	0.90	**22.95 ± 3.36**	3.80	38.40 ± 6.23	1.52	30.91 ± 5.30	1.26	**79.88 ± 4.07**	6.82	94.04 ± 0.41	20.83 s
PFV	**97.14 ± 1.01**	0.26	**26.74 ± 1.84**	3.54	**47.20 ± 3.90**	1.39	**34.18 ± 5.36**	0.74	**80.48 ± 5.51**	9.23	**94.19 ± 0.73**	5.97 s

2Wd, 2-Wasserstein distance; MK, multiscale kernel; SWK, sliced Wasserstein kernel; PL, persistence landscape; PI, persistence image; RM, Riemmanian manifold; PBoW, persistence bag of words; wPBoW, weighted persistence bag of words; sPBoW, stable persistence bag of words; PVLAD, persistence VLAD; sPVLAD, stable persistence VLAD; PFV, persistence Fisher vector

Table 3 Results of EXP-B averaged over 5 runs

DESCR. NAME	Motion Capture (5 REPS.)		GeoMat (5 REPS.)		Reddit-5k (5 REPS.)		Reddit-12k (5 REPS.)		PetroSurf3D (4 REPS.)	
	Score	Time (s)	Score	Time (s)	Score	Time (s)	Score	Time (s)		
PI	**92.00 ± 5.06**	166.59	22.45 ± 0.00	7243.12	49.34 ± 2.83	4759.66	**38.37 ± 0.89**	11,329.96	**80.40 ± 4.92**	12,,137.04 s
approxPI	**92.67 ± 4.35**	6.68	26.92 ± 0.00	8.99	**51.14 ± 3.12**	3.85	**37.66 ± 0.66**	8.28	**78.96 ± 5.14**	**13.71 s**
RM	**94.00 ± 2.79**	15.50	9.37 ± 0.00	222.59	46.25 ± 3.11	108.12	32.55 ± 1.08	215.50	**79.85 ± 5.01**	1450.95 s
PBoW	**94.00 ± 3.65**	**1.17**	28.96 ± 0.40	**5.24**	**49.94 ± 3.28**	**1.53**	**38.64 ± 0.87**	**4.78**	**80.34 ± 5.25**	28.52 s
wPBoW	**94.67 ± 2.98**	2.48	**29.79 ± 0.32**	26.05	48.90 ± 2.66	13.57	36.97 ± 1.03	20.75	**80.43 ± 5.35**	82.96 s
sPBoW	89.33 ± 4.35	7.60	27.79 ± 0.75	29.70	46.57 ± 3.33	5.17	35.34 ± 1.43	29.34	**79.91 ± 5.17**	161.76 s
PVLAD	**94.67 ± 2.98**	1.37	23.78 ± 0.66	28.24	42.57 ± 2.83	14.35	30.90 ± 0.31	22.75	**78.43 ± 4.84**	80.16 s
sPVLAD	**93.33 ± 3.33**	7.31	27.40 ± 0.64	26.81	42.73 ± 2.26	31.39	33.47 ± 1.01	35.49	**80.22 ± 5.24**	122.88 s
PFV	**94.00 ± 3.65**	3.02	**29.29 ± 0.61**	27.91	49.10 ± 1.91	15.38	**39.21 ± 1.30**	21.38	**80.68 ± 5.16**	83.00 s
State of art	89.70[a]		22.32 ± 0.76[b]		49.10[c]		38.50[d]		n/a[u]	

2Wd, 2-Wasserstein distance; MK, multiscale kernel; SWK, sliced Wasserstein kernel; PL, persistence landscape; PI, persistence image; RM, Riemmanian manifold; PBoW, persistence bag of words; wPBoW, weighted persistence bag of words; sPBoW, stable persistence bag of words; PVLAD, persistence VLAD; sPVLAD, stable persistence VLAD; PFV, persistence Fisher vector

[a] Ali et al. (2007)

[b] Results obtained using original classification algorithm of DeGol et al. (2016) fed with only surface shape information (i.e. map and histogram of normals)

[c] Hofer et al. (2017)

[d] Not applicable, original research Zeppelzauer et al. (2017) use dice similarity index (DSC) instead of accuracy

Table 4 Parameters tested for EXP-A

Descr.[a]		Synthetic	GeoMat	Reddit-5k	Reddit-12k	PetroSurf3D	3D Shape Segm.
MK	n	{**0.5**, 1, 2}	{0.5, **1**, 2}	{**0.5**, 1, 2}	{0.5, 1, **2**}	{**0.5**, 1, 2}	{**0.5**, 1, 2}
SWK	n	{**50**, 100, 150, 200, 250}	{**50**, 100, 150, 200, 250}	{**50**, 100, 150, 200, 250}	{**50**, 100, 150, 200, 250}	{**50**, 100, 150, 200, 250}	{**50**, 100, 150, 200, 250}
PI/approxPI	r	{10,20,...,**50**,...100}	{10,20,40,60,80,**100**}	{10,20,30,40,50,60,80,100,**120**}	{10,20,30,**40**,50,60,80,100,120}	{10,20,...**80**,...100}	{10,20,...**70**,...100}
	σ	{**0.1**, 0.5, 1, 2}	{**0.5**, 1, 2}	{**0.5**, 1, 2, 3}	{**0.5**, 1, 2, 3}	{0.5, **1**, 2}	{**0.5**, 1, 2, 3}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
RM	r	{**10**, 20, 40}	{**10**, 20, 40, 60}	{**10**, 20, 40, 60}	{10, 20, **40**, 60}	{10, 20, 40, **60**}	{**10**, 20, 40, 60}
	σ	{0.1, **0.2**, 0.3}	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}
	d	{**50**, 75, 100}	{50, **75**, 100}	{50, **75**, 100}	{50, 75, **100**}	{50, **75**, 100}	{**50**, 75, 100}
PBOW	N	{10,20,30,...**200**}	{10,20,**30**,40,60,80,...200}	{10,20,**30**,...,60,80,100,...200}	{10,20,30,40,**50**,60,80,100,120}	{10,20,30,40,50,60,80,...**160**, ...200}	{10,20,...,**80**,...100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{1000, 5000, **10,000**}	{**2000**, 10,000, 50,000}	{**2000**, 10,000, 50,000}	{5000, **10,000**, 50,000}	{**5000**, 10,000, 50,000}	{5000, **10,000**, 20,000}
wPBOW	N	{10,20,...**110**,...200}	{10,20,30,40,60,...,**160**,...200}	{10,**20**,...,60,80,100,...200}	{10,20,30,40,50,60,**80**,100,120}	{10,20,30,40,**50**,60,80,...200}	{10,20,...**60**,...00}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**1000**, 5000, 10,000}	{2000, **10,000**, 50,000}	{**2000**, 10,000, 50,000}	{**5000**, 10000, 50,000}	{**5000**, 10,000, 50,000}	{5000, **10,000**, 20,000}
sPBOW	N	{10,20,...**140**,...200}	{10,20,**30**,40,60,80,...200}	{10,**20**,...,60,80,...200}	{**10**,20,30,40,50,60,80,100,120}	{10,20,30,40,50,60,80,**100**...200}	{10,20,...**70**,...100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{1000, **5000**, 10000}	{**2000**, 10000, 50000}	{2000, **10,000**, 50,000}	{**5000**, 10,000, 50,000}	{5000, 10,000, **50,000**}	{5000, 10,000, **20,000**}
PVLAD	N	{**10**,20,...200}	{**10**,20,30,40,60,80,...200}	{**10**,20,...,60,80,100,...200}	{**10**,20,30,40,50,60,80,100,120}	{10,20,**30**,40,50,60,80,...200}	{10,**20**,...,100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{1000, 5000, **10,000**}	{**2000**, 10,000, 50,000}	{2000, **10,000**, 50,000}	{5000, **10,000**, 50,,000}	{5000, **10,000**, 50,000}	{5000, 10000, **20,000**}
sPVLAD	N	{10,20,**30**,...200}	{10,**20**,30,40,60,80,...200}	{10,20,**30**,...,60,80,100,...200}	{**10**,20,30,40,50,60,80,100,120}	{10,20,30,40,**50**,60,80,...200}	{10,20,...,**100**}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{1000, **5000**, 10,000}	{2000, 10,000, **50,000**}	{**2000**, 10,000, 50,000}	{5000, **10,000**, 50,000}	{5000, **10,000**, 50,000}	{5000, 10,000, **20,000**}
PFV	N	{10,**20**,...200}	{10,**20**,30,40,60,80,...200}	{**10**,20,...,60,80,100,...200}	{10,20,30,**40**,50,60,80,100,120}	{10,20,**30**,40,50,60,80,...200}	{10,**20**,...,100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**1000**, 5000, 10,000}	{2000, 10,000, **50,000**}	{**2000**, 10,000, 50,000}	{5000, **10,000**, 50,000}	{5000, 10,000, **50,000**}	{5000, 10,000, **20,000**}

The optimal parameters are in bold

n, number of lines slicing the plane in SWK; r, resolution of PI/approxPI or density map in RM; σ, sigma of the Gaussians employed; W, weighting (either no weighting (NW) or with weighting (W); d, dimension of Principal Geodesic Analysis on the hypersphere; N, the number of codewords of persistence codebooks datasets in our study it is inconvenient to compute the explicit kernel matrices for the kernel-based approaches for computational reasons. Nevertheless, to still enable a fair comparison of all approaches, we sub-sample the datasets in EXP-A to reduce their size, i.e. by randomly selecting 30 (GeoMat), 100 (reddit-5k), 50 (reddit-12k) and 390 (PetroSurf3D) samples for each class. In EXP-B we solely evaluate the vectorized representations on the datasets as described in Sect. 4.1.

The evaluation procedure for each dataset is as follows. For the synthetic dataset, we subsample 80% of the samples as training data and use the remaining samples for testing. For GeoMat dataset, we use the original train/test partition with 400/200 samples per class. For Reddit experiments, we employ the original ratio of 90% graphs in the training set and the remaining 10% in the test set. For the 3D shape segmentation dataset, we employ the original 50/50 split. The train/test split ratio of the motion capture dataset is 80/20. For all datasets, we average the achieved performance of grid search over 5 repetitions of random selected training and test partitions. The only exception is PetroSurf3D, where we divided the set of all surfaces into 4 folds (resulting in four repetitions) according to original work of Poier et al. (2017).

Ultimately, we run a Wilcoxon signed-rank test (with p value of 0.1) on the results to identify which results significantly differ from the best obtained result, and which ones do not, and can thus be considered equally good. The comparison is performed between the best method (the one with the best mean accuracy) and all the other methods, for each experiment separately. The mean accuracy is obtained as an average over 5 runs with the same train/test divisions used by all compared methods. The number of repetitions is relatively small for statistical tests, therefore we set p value to 0.1.

The entire code of all experiments (implemented in Matlab) is available at https://github.com/bziiuj/pcodebooks. For external approaches, we use the publicly available implementations of the original authors. For clustering and bag-of-words encoding, we employ the VLFeat library (Vedaldi and Fulkerson 2008).

5 Results

Table 2 summarizes the results obtained in our experiments for EXP-A and EXP-B. For each combination of dataset and approach, we provide the obtained classification accuracy (including the standard deviation) and the processing time needed to construct the representations (excluding the time for classification). Note that for the synthetic dataset and the 3D shape segmentation dataset, results of EXP-A and EXP-B are equal, as no subsampling was needed to perform EXP-A.

In all experiments, codebook representations of persistence diagrams can compete with or even outperform the compared approaches. From EXP-A we further observe that vectorized representations (including the proposed ones), in general, perform better than kernel-based approaches. In case of the PetroSurf3D dataset, it is impossible to unambiguously determine the best method, since all approaches work equally well. For all other datasets, only the 2-Wasserstein distance and the sliced Wasserstein kernel attain accuracy comparable to vectorized approaches. Among the compared vectorized representations, PI in most cases outperforms RM and will thus serve as the primary approach for comparison with our approaches in subsequent sections. When comparing the stable vs. unstable variants of PBoW and PVLAD, we observe that PBoW in most cases outperforms its stable equivalent, especially for motion capture database in EXP-B, where sPBoW is ~5% worse than

Table 5 Parameters tested for EXP-B

Descr.[a]		Motion capture	GeoMat	Reddit-5k	Reddit-12k	PetroSurf3D
PI/approxPI	r	{**10**,20,30,40,50,60}	{10,20,30,40,50,**60**}	{10,20,30,40,50,**60**}	{10,20,30,40,**50**,60}	{10,20,30,40,**50**,60}
	σ	{0.5, 1, **2**}	{**0.5**, 1, 2, 3}	{**0.5**, 1, 2}	{**0.5**, 1, 2}	{0.5, 1, **2**}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
RM	r	{**20**, 40, 60}	{**10**, 20, 40, 60}	{**20**, 40, 60}	{**20**, 40, 60}	{20, **40**, 60}
	σ	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}	{**0.1**, 0.2, 0.3}	{0.1, 0.2, **0.3**}
	d	{25, 50, 75, **100**}	{**50**, 75, 100}	{**25**, 50, 75, 100}	{**25**, 50, 75, 100}	{25, 50, **75**, 100}
PBOW	N	{10,20,...50,60, **80**,100}	{10,20,40,60,80,**100**,...200}	{10,20,...60,80,**100**...200}	{10,20,...50,60,80,**100**}	{10,20,...**50**,60,80,100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**2000**, 5000, 10,000}	{2000, 10,000, **50,000**}	{**2000**, 10,000, 50,000}	{5000, 10,000, **50,000**}	{**5000**, 10,000, 50,000}
wPBOW	N	{10,20,...**50**,60, 80,100}	{10,20,40,60,**80**,...200}	{10,20,...**50**,60,80,...200}	{10,20,...50,60,80,**100**}	{10,20,...50,60,80,**100**}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**2000**, 5000, 10000}	{**2000**, 10,000, 50,000}	{**2000**, 10,000, 50,000}	{**5000**, 10,000, 50,000}	{**5000**, 10,000, 50,000}
sPBOW	N	{**10**,20,...50,60, 80,100}	{10,**20**,40,60, ...200}	{10,20,**30**...60,80,...200}	{10,20,...50,60,80,**100**}	{10,20,...50,60,80,**100**}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{2000, 5000, **10,000**}	{**2000**, 10,000, 50,000}	{2000, **10,000**, 50,000}	{5000, **10,000**, 50,000}	{5000, **10,000**, 50,000}
PVLAD	N	{10,20,...50,**60**, 80,100}	{10,20,**40**,60, ...200}	{10,20,...60,80,**100**...200}	{10,20,...50,60,80,**100**}	{10,20,**30**,40,50,60,80,100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**2000**, 5000, 10,000}	{2000, **10,000**, 50,000}	{2000, 10,000, **50,000**}	{5000, **10,000**, 50,000}	{**5000**, 10,000, 50,000}
sPVLAD	N	{10,20,30,...50,60, 80,**100**}	{**10**,20,40,60, ...200}	{10,20,...60,80,...**160**,...200}	{10,20,...50,60,**80**,100}	{10,20,...50,60,**80**,100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**2000**, 5000, 10,000}	{2000, **10,000**, 50000}	{2000, 10,000, **50,000**}	{5000, 10,000, **50,000**}	{5000, 10,000, **50,000**}
PFV	N	{10,20,...50,60, 80,**100**}	{**10**,20,40,60, ...200}	{10,20,**30**,40,50,60,80,...200}	{**10**,20,...50,60,80,100}	{10,20,30,**40**,50,60,80,100}
	W	{W, NW}	{W, NW}	{W, NW}	{W, NW}	{W, NW}
	S	{**2000**, 5000, 10000}	{2000, **10,000**, 50,000}	{2000, 10,000, **50,000**}	{5000, 10,000, **50,000**}	{5000, 10,000, **50,000**}

the other methods. Opposite is the case for PVLAD, where sPVLAD in most cases yields a higher performance. Nevertheless, sPVLAD, for almost all datasets, is still significantly worse than any other codebook variant. The wPBoW works almost as well as non-weighted PBoW. Only in GeoMat and Reddit12K in EXP-B there exist visible differences in performance; however, even these results are very close (~1% and ~1.5%). PFV variant seems to be the best approach, since in all experiments we can find it among the best performing methods. Overall, however, PBoW versus PFV, there is no clear winner.

Large differences exist in the processing times of the different approaches. The highest runtimes are obtained for the kernel-based approaches going up to 63k seconds for the PetroSurf3D dataset. The slowest kernel is 2Wd followed by MK, and approximately one order of magnitude faster PL and SWK. Note that computation complexity depends not only on a number of persistence diagrams, but is also highly affected by the average number of points per diagram. For the vectorized approaches, PI takes longest to compute. The runtimes, however, vary strongly, depending on the resolution of the employed PI (note that we have estimated the optimal parameters for each dataset by a grid search over all hyperparameters, see Tables 4 and 5). The RM representation is one to two magnitudes faster than PI.[17] The proposed approaches outperform almost all evaluated approaches in runtime for all datasets, both for EXP-A and EXP-B. The gain in runtime efficiency ranges from one to up to four orders of magnitude. For the largest dataset in the experiments (PetrSurf3D in EXP-B), the fastest (PBoW) and the slowest (sPBoW) codebook approaches are still 3 and 2 magnitudes faster than PI, while reaching comparable accuracy. Competitive runtimes can be achieved by the approximated version of PI (approxPI). However, this leads to a slight drop in performance for most datasets compared to the exact (and stable) implementation of PI. Moreover, we observed that approxPI performs much worse than our approaches in the case of 3D Shape Segm., the largest database of EXP-A. Detailed investigation on this performance revealed that it is primarily dominated by kernel computations (reported in EXP-A due to comparison with kernel methods). It is expected, as the size of approxPI in the experiment mentioned above is 4900, while the size of sPBoW is only 70. Concerning EXP-A, the codebook approaches are comparable in computing time, which is due to the small size of the datasets. EXP-B demonstrates well how the different approaches scale to larger data. It shows that PBoW scales best (is fastest) and still obtains optimal results in all but one case. It thus represents the best tradeoff between time efficiency and classification accuracy. See Sect. 5.2 for further discussion.

From our experiments we conclude that persistence codebooks are significantly faster than most approaches while achieving similar or even better performance level. This shows that the codebooks capture well the essential information contained in the PDs and important for the respective classification tasks. The variablity of runtimes between the different codebook variants is low compared to the other approaches. Thus, for the selection of the appropriate codebook approach for a given problem in practice, the runtime plays a secondary role.

In the following sections, we analyze selected aspects of the novel representations in greater detail, such as runtime, dependency on parameters and the scalability of the approach to large number of input PDs.

5.1 Accuracy versus codebook size and weighted sub-sampling

The most important parameter for codebook-based representations is the codebook size N, i.e. the number of clusters. There is no commonly agreed analytic method to estimate the

[17] Note that for both representations we use the implementations provided by the original authors.

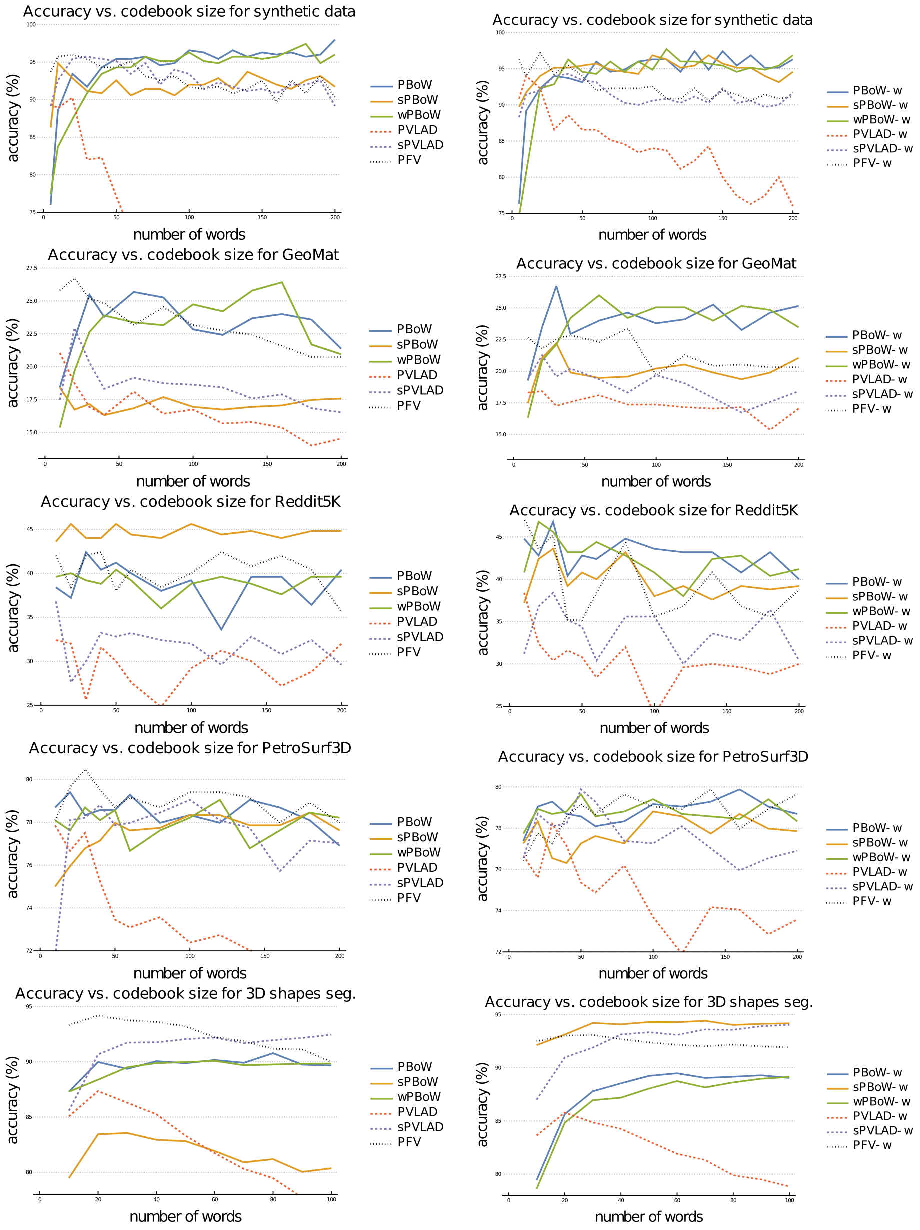

Fig. 4 Accuracy vs. size of a codebook for datasets from EXP-A without (left column) and with codebook weighting (right column) in codebook generation

optimal codebook size; thus, the estimation is usually performed empirically. To investigate the sensitivity of codebook approaches and their performance on the codebook size, each approach was evaluated for a sequence of N values (see Tables 4, 5). The results are presented in Fig. 4, both without (left column) and with weighted sub-sampling (right column) of the consolidated PD.

We can observe that all three variants of PBoW (PBoW, wPBoW and sPBoW) reach optimal performance at the level of about 50 words in a codebook (or earlier), a further increase of codebook size does not necessarily further improve its efficiency. In some cases, there is a slight improvement (synthetic data), in other cases performance goes down slightly (GeoMat) or remains a the same level. This shows that the codebook size is not rather insensitive parameter, once a certain minimum size is surpassed.

The remaining approaches (PVLAD, sPVLAD and PFV) show a general tendency to achieve their best performance early, with codebooks containing less than 40 words, and after that the accuracy drops substantially. The most prominent example of this behavior is depicted by the PVLAD method. It is caused by the fact that in cases where there is just a few clusters, it is simpler to capture well both the zeroth and the first moments, because clusters occupy large regions. However, once the number of clusters gets larger, cluster size shrinks and the assignment gets unstable.

Figure 4 further shows the effect of weighting during sub-sampling for codebook generation. This can be best observed from the performance curves of sPBoW, where the effect is largest. For 3D shapes, weighting leads to a dramatic improvement in accuracy. For for synthetic data, GeoMat and PetroSurf3D, there is also a moderate improvement in performance. Only for Reddit5K, weighted subsampling degrades performance. For other methods, however, the introduction of weighted subsampling does yield an improvement on the Reddit5K experiment (see e.g. PBoW, wPBoW and sPVLAD). In the majority of cases, weighted subsampling has a positive impact on performance. For PVLAD, weighted subsampling even seems to compensate for weaknesses of the representation in situations where codebook sizes are large.

Overall, we conclude, that optimal and universal choice for codebook size is about 50 in case of PBoW, wPBoW and sPBoW; while for the remaining methods, 20 words seems to be sufficient. These values are thus good starting points for hyperparameter optimization on other datasets. The choice of weighted vs. non-weighted subsampling seems to be dataset dependent. For 3D shapes, for example, strong performance gains are achieved. For the other datasets the trend is not so clear.

5.2 Accuracy versus time

Tables 2 and 3 show that our approaches achieve comparable performance on almost all of the evaluated datasets and partly even outperform the compared approaches. Additionally, they beat all methods in speed. While the above tables show results for the optimal parameters (from the classification accuracy point of view), we decided to analyze the relationship between accuracy and computation time. For this purpose, we use PI and approxPI as references for comparison, because they represent the strongest competitors (in the sense of accuracy) of the proposed representations.

In Fig. 5, we plot accuracy vs. time for the proposed approaches and PI for all datasets from EXP-B. We decided to focus on EXP-B here, because it operates on larger datasets than EXP-A and is thus better suited to study runtime efficienty. We vary the parameters with most influence placed on runtime (codebook size for persistent codebooks as well as the resolution of PI and approxPI) according to the values provided in Table 5. This directly influences the output dimension of the representation and is reflected by the area of the circles in Fig. 5, i.e. larger diameter means higher dimension. Note that experiments

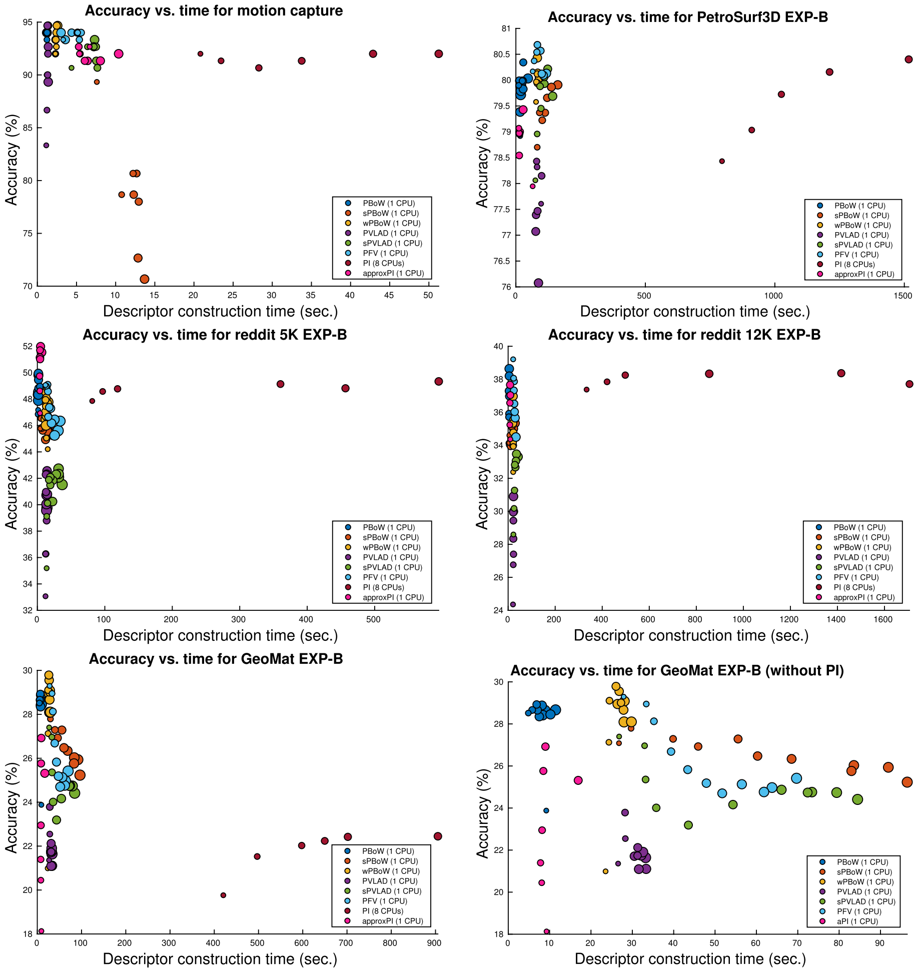

Fig. 5 Accuracy vs. time for codebook approaches compared with PI and approxPI (the strongest related representations) applied to all datasets from EXP-B. The size of colored points represents the size N of codebooks or the resolution r of PI and approxPI (evaluated values for N and r are those listed in Table 5). Note that the actual times of computation for the construction of the representations are presented. Codebooks and approxPI were computed on 1 CPU, while PI was constructed by using 8 CPUs in parallel. Moreover, SVM training and prediction time was not taken into consideration. This would further increase computational times, especially for PI and approxPI due to their larger dimension. The bottom-right plot shows results for codebook approaches and approxPI (but skipping PI) in the case of GeoMat dataset for a narrower range of x-axis (for better visibility)

with codebooks and approxPI were performed on 1 CPU, while experiments on PI were performed in parallel on 8 CPUs. Therefore, the total runtime differences are in fact even larger than depicted. For more compact visualization (and avoiding a logarithmic scale which would compress too much), we decided not to take the number of CPUs into account for plotting. We can see clearly that the runtime of PI is always significally larger than any codebook representation. The accuracy obtained varies. For all datasets the performance level of PI is reached (or even superseeded) much quicker. In the case of GeoMat dataset, codebooks clearly outperform PI (while consuming much less time); and in case of the other experiments, they quickly achieve a similar performance level. The computational cost of achieving a higher performance with PI is over-proportionally high, while the performance gain is actually rather limited (approx. +1%). Interestingly, approxPI strongly outperforms PI and can compete well with our representations in terms of runtime. The computation time of approxPI is in the same order of magnitude as that of the persistence codebook representation. However, the performance is in most cases lower as can be seen in Fig. 5 where pink dots represent approxPI. Note that for better visibility, we provide a wider and a narrower range of x-axis for the performance comparison of GeoMat dataset (last row of Fig. 5) because the latter allows easier comparison between the codebook approaches. Among our methods, PBoW is the fastest. We observe that runtimes of PBoW, wPBoW and PVLAD are almost not affected by codebook size. The other approaches, i.e. sPBoW, sPVLAD and PFV, that involve the computation of Gaussians, clearly require much more time when codebook size is increased. However, as observed before, larger codebook sizes are not necessarily required to obtain good accuracy, which mitigates the situation. approxPI is equally fast as PBoW but is not able to achieve the same level of performance.

5.3 Time versus dataset size

To investigate the runtime behavior of the proposed approaches in more detail, we evaluate how they scale to increasing dataset sizes (i.e. increasing numbers of input PDs). To this end, we employ the largest dataset in our experiments (PetroSurf3D) and randomly sample different numbers of PDs, starting from 1000 to 10, 000 in steps of 1000. To get a detailed breakdown of computation time, we separately measure the time needed for codebook generation, histogram assignment, and classification. The computation of the PDs is the same for all approaches, and thus is not included in this breakdown.

From the results presented in Fig. 6, we conclude that runtime grows almost linearly with dataset size. For the approaches *without* weighted subsampling (upper part in Fig. 6), most of the computation time is spent on histogram assignment and classification. Histogram assignment takes more time for more complex encoding methods, such as PVLAD and PFV. In case of PBoW, histogram assignment is particularly fast because of k-d trees being used (Bentley 1975). For sPBoW, Gaussian likelihood has to be computed, which slows down the computation. Assignment time, however, grows linearly with dataset size. Classification time takes the major part for PVLAD and PFV. This is due to the fact that the computational complexity of both, primal and dual SVM optimization, depends on dimensionality (Chapelle 2007), which is higher in case of PVLAD and PFV. The distribution of computation times is similar for the persistence codebook approaches *with* weighted subsampling (lower part in Fig. 6).

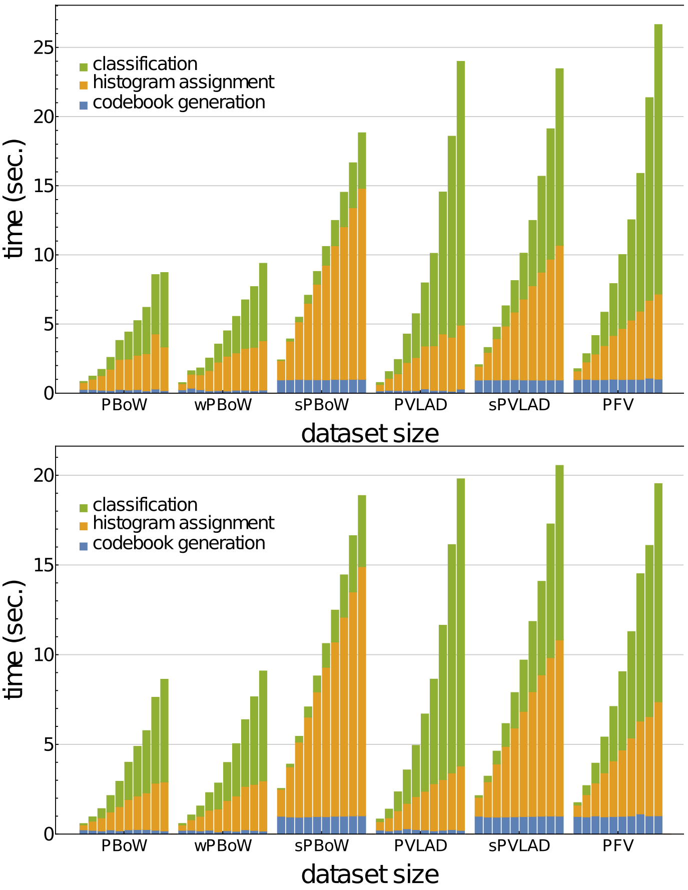

Fig. 6 Time versus dataset size for all proposed persistent bag-of-words approaches (all with codebook size $N = 50$). We measure the time needed for codebook generation (blue, bottom), histogram assignment (orange, middle), and classification (green, top), separately. Every consecutive bar represents an increasingly growing number of samples, from 1000, 2000, ..., 10000. Upper figure shows results for unweighted methods, while those for the weighted versions are presented in the bottom. (Color figure online)

5.4 Qualitative analysis

In this section, we investigate PBoW with a special focus on its discriminative abilities. For this purpose, we employ the synthetic dataset as a proof-of-concept and GeoMat (for which we outperform other representations by a large margin) to investigate how this performance increase is achieved by PBoW compared to related approaches.

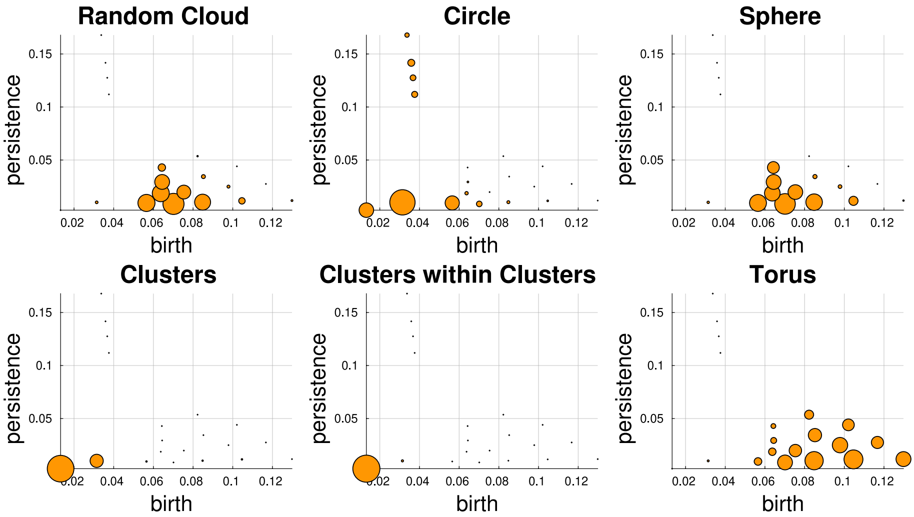

Fig. 7 Average codebook histograms computed for each of the six shape classes of the synthetic dataset. The cluster center of each codeword is presented as a circle in the birth-persistence domain. The area of the circles reflects the histogram values of the specific class. For all classes, the same codebook (same clustering) is employed; thus, dot locations are the same on all plots. The differences between the circles reflect the class differences

5.4.1 Synthetic dataset

We compute PBoW with $N = 20$ clusters for the synthetic dataset and visually analyze the codeword histograms obtained by (hard) assignment. To this end, for each of the six shape classes, we compute the average codebook histogram (over all samples of each class) to obtain one representative PBoW vector per class. The averaged PBoW histograms for all classes are presented in Fig. 7. Instead of only providing the histograms themselves, for each codeword of the histogram we plot the corresponding cluster center as a circle in the original birth-persistence domain and encode the number of assigned codeworks (the actual values of the histograms) in the area of the circles, i.e. the larger the count for a cluster, the larger the circle. The advantage of this representation is that the spatial distribution of the codewords in the PD is preserved.

From Fig. 7 we can see that, except for the classes “random cloud” and “sphere” (which are difficult to differentiate), all classes generate strongly different cluster distributions. Class “circle”, for example, uniquely activates four clusters with strong persistence (top-left corner) and the “torus” class distributes its corresponding code words across a large number of clusters representing less persistent components.

Figure 7 further illustrates an important property of persistence bag-of-words, namely its sparse nature. More specifically, areas with no points in the consolidated persistence diagram will contain no codewords (clusters). In Fig. 7, for example, no codeword is

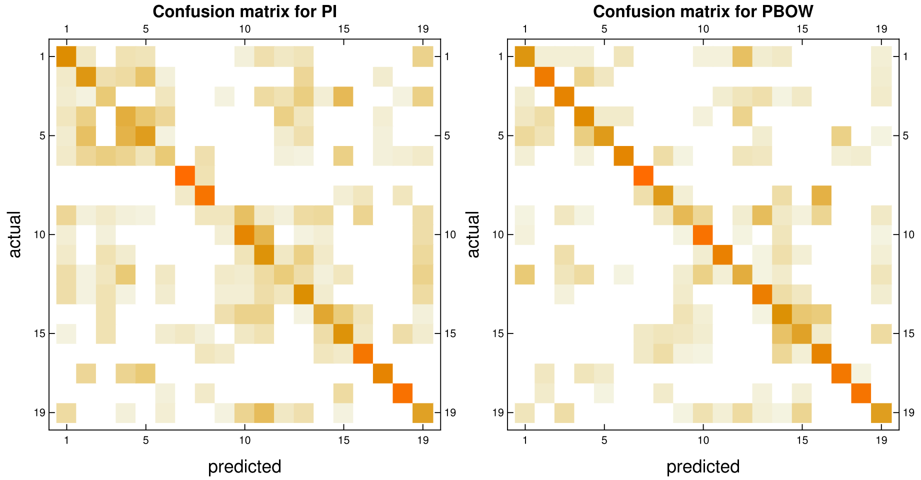

Fig. 8 Confusion matrix for PI (left) and PBoW (right) on the GeoMat dataset from EXP-A. From the diagonal of the matrices we can see that PBoW outperforms PI for many classes (e.g. classes 2–5, 9 and 12). Furthermore, there are less confusions (off-diagonal values) for PBoW

obtained in the upper-right quadrant of the diagram, since no components are located there for the underlying data. Therefore, these unimportant areas are neglected and not encoded into the final representation. This not only reduces the dimension of the final representation, but further makes the representation adaptive to the underlying data. This in turn increases the information density in the obtained representation.

5.4.2 GeoMat dataset

We further investigate the performance on the GeoMat dataset to explain why (s)PBoW outperforms PI, approxPI and RM by such a large margin (see Table 2). To this end, we generate confusion matrices for PI and PBoW (see Fig. 8) to investigate their discriminative abilities. The matrices show that PBoW, for example, achieves better discrimination between classes “cement smooth” and “concrete cast-in-place” (i.e. classes 4 and 5). Average PBoW histograms for those classes are shown in Fig. 9. The histograms are on the first sight similar (upper row in Fig. 9). However, by zooming-in towards the birth-persistence plane in Fig. 9 (bottom row), differences become better visible. The plots in the center illustrate the difference between the class distributions (red color means left class is stronger, blue means right class is stronger for this cluster). The classes differ by fine-grained spatial differences. The set of three blue points around birth time of 0 (which are characteristic for class “concrete cast-in-place”) surrounded by red points (which are characteristic for class “cement smooth”) illustrates this well (see lower central plot). For the

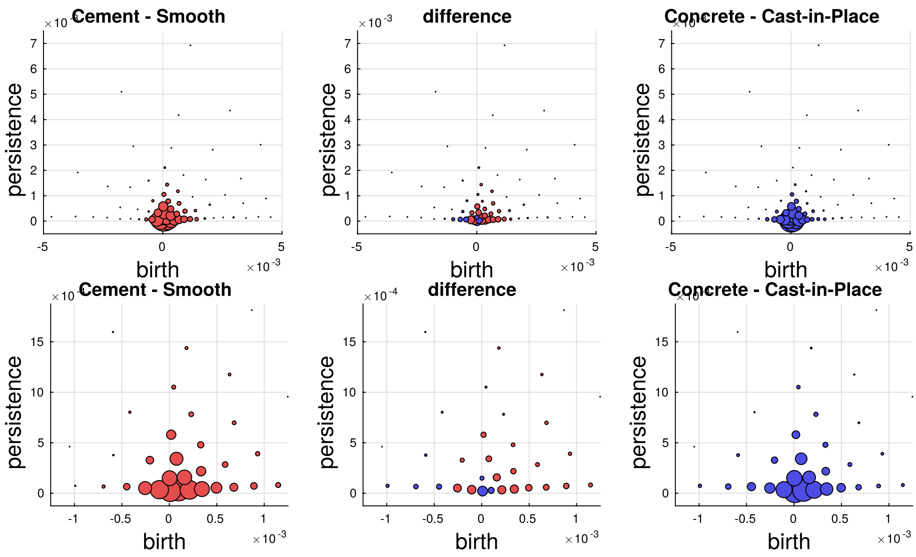

Fig. 9 Comparison of averaged PBoW histograms for class “cement smooth” (left, red) and “concrete cast-in-place” (right, blue) from GeoMat dataset (top row: total view; bottom row is zoomed in). The plot in the center shows the difference between the classes, where red color means that the left class has stronger support for this cluster and blue means that the right class has stronger support. The classes differ by fine-grained spatial differences, which are not distinguishable in other vectorized representations. (Color figure online)

discrimination of these two classes, a particularly fine-grained codebook with many clusters is needed. PI (and approxPI) have problems with such fine-grained structures, because due to its limited resolution, all topological components in the most discriminative area would most likely fall into one PI-pixel. Therefore, an extraordinary high resolution would be necessary to capture the discriminative patterns between those two classes. The bag-of-words model makes our approaches independent of the resolution and enables to capture even fine differences adaptively and in an unsupervised way. In Fig. 10 we show a similar comparison for classes “brick” and “concrete cast-in-place” (i.e. classes 2 and 5).

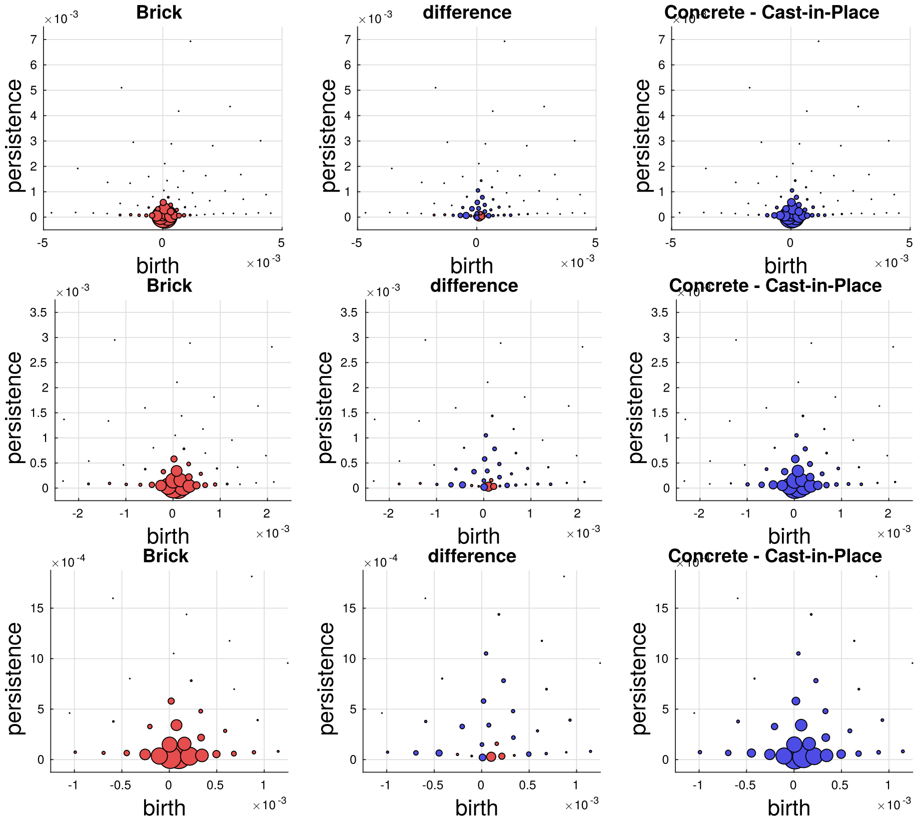

Fig. 10 Comparison of averaged PBoW histograms for classes “brick” (left, red) and “concrete cast-in-place” (right, blue) from GeoMat dataset (top row: total view; 2nd row: zoomed in view; 3rd row: even further zoomed in view). The plot in the center shows the difference between the classes, where red color means that the left class has stronger support for this cluster and blue means that the right class has stronger support. The classes differ by fine-grained spatial differences, which are not distinguishable in other vectorized representations. (Color figure online)

6 Conclusion

We have introduced the concept of persistence codebooks, a novel fixed-length vectorial representation for persistence diagrams. Persistence codebooks employ bag-of-words encodings to quantize the persistence diagram into a vectorized representation. We propose different types of encodings (based on traditional bag-of-words, VLAD and Fisher Vectors), investigate their theoretic properties, such as their stability with respect to 1-Wasserstein, and introduce robust variants of the representations. Experiments on seven heterogeneous datasets show that they consistently achieve comparable performance to related methods, and partly even outperform them, with significantly shorter computation time. Though there is no overall winner among the introduced representations, we conclude that PFV is a powerful representation, as it achieves peak performance over all evaluated datasets. It is followed by the PBoW variants, which also consistently achieve peak performance (but not for all datasets). PVLAD cannot compete with the other representations in our experiments and is thus less recommended. Moreover, we observe a certain tradeoff between computation time and accuracy when comparing stable and non-stable representations. Unstable representations like approxPI and PBoW are particularly fast. However, in general, they cannot compete with stable descriptors like PFV in terms of accuracy.

The novel representations have both attractive theoretic properties as well as practical properties, i.e. compactness, expressiveness, as well as the ability to adapt to the inherent sparsity of persistence diagrams. They can be constructed in a completely unsupervised fashion and achieve a high discriminativity compared to related approaches. The high computational efficiency of persistence codebooks could in future facilitate the application of TDA to larger datasets than possible today and enable real-time applications.

Acknowledgements The work presented in this article was supported by the National Science Centre, Poland under grant no. 2015/19/D/ST6/01215 and by the Austrian Research Promotion Agency (FFG) under project no. 856333 and no. 866880. We acknowledge the support of Dioscuri program initiated by the Max Planck Society, jointly managed with the National Science Centre (Poland), and mutually funded by the Polish Ministry of Science and Higher Education and the German Federal Ministry of Education and Research.

Appendix: Background on persistent homology

In this section we present basic introduction to persistent homology. Please consult (Edelsbrunner and Harer 2010; Edelsbrunner et al. 2002; Zomorodian and Carlsson 2005) for more information.

Topological spaces are typically infinite objects and, for the sake of data analysis, they have to be finitely represented by simplified objects called *cell complexes*. Cell complexes are build from *cells*: topologically simple objects having the property that an intersection of every pair of cells is either empty, or contains yet another cell in the cell complex.

A *simplicial complex* is a particular instance of a general cell complex. It is a natural tool in the study of multi-dimensional point cloud data. Cells of simplicial complex are called *simplices* and, in this particular case, are formed with convex hulls of collections of nearby points in the point cloud. Simplices are uniquely characterized by a collection of points involved in their convex hulls. A simplicial complex $\mathcal{X}$ needs to satisfy the following property: for every pair of simplices $\sigma, \tau \in \mathcal{X}$, $\sigma \cap \tau$ is either empty or a simplex in $\mathcal{X}$.

Given a point cloud X with a distance or a similarity measure d and a parameter $r > 0$, one can define a *Vietoris-Rips complex* $VR(X, r)$. It is a simplicial complex whose every simpliex $\sigma = \{v_0, v_1, \dots, v_k\}$ satisfies $d(v_i, v_j) \leq r$ for every $i,j \in \{0, \dots, k\}$. For every simplex $\sigma \in VR(X, r)$, one can define a diameter of σ being the largest distance

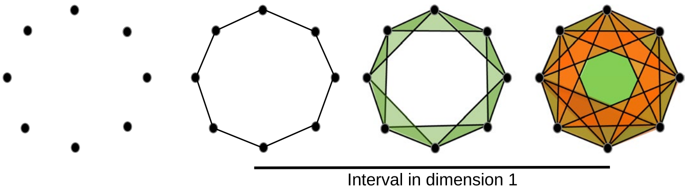

Fig. 11 Various stages of a construction of a Vietoris-Rips complex for eight points sampled from a circle. Initially, for sufficiently small radius, only vertices are present in the complex. Gradually, more and more edges along with higher dimensional simplices of an increasing diameter are added. In all but the initial and final stage of the construction the topology of a circle is visible, and therefore will be recovered by PH in dimension one (depicted by the long bar below the picture)

between the points in σ. This gives a natural ordering of simplices in $VR(X, r)$: primarily by diameter of simplices and secondarily (when diameters of two simplices are the same) by inverse of the number of points in simplices[18]. It is easy to see that every prefix of such an ordering forms a simplicial complex, and therefore any increasing sequence of numbers $0 < r_1 < r_2 < \ldots < r_n$ yields a nested sequence of simplicial complexes:

$$\emptyset \subset X = VR(X, 0) \subset VR(X, r_1) \subset$$
$$VR(X, r_2) \subset \cdots \subset VR(X, r_n)$$

Another typical scenario when such a nested sequence of cell complexes arises is the case of values of a function f discretized on a grid G. The function $f : G \to \mathbb{R}$ is typically an output of some numerical method. The grid G naturally corresponds to cubical complex $\mathcal{G}$, and the function f provides an ordering of maximal cubes in the complex. This ordering induces a nested sequence of cubical complex, very much like a nested sequence of Vietoris-Rips complexes discussed above.

To cover those and other possible cases, later in this section we will focus on a general case of filtered cell complex:

$$\emptyset = \mathcal{C}_0 \subset \mathcal{C}_1 \subset \cdots \subset \mathcal{C}_n = \mathcal{C},$$

keeping in mind that most typically it will come from a point cloud, or numerical simulations on a grid.

Having a complex $\mathcal{C}_i$ in the filtration, one can define its homology, $H(\mathcal{C}_i)$. Rather than providing a formal definition, which can be found in Edelsbrunner and Harer (2010), we will focus on the intuitive understanding of the concept. Homology in dimension 0 measures number of connected components. In dimension 1 it measures the cycles, which do not bound to a (deformed) surface. In dimension 2 it corresponds to voids, i.e. regions of space totally bounded by a collection of triangles (very much like a ball bounds the void inside it). The idea of a cycle bounding a hole in the complex can be formalized using homology theory for arbitrary dimension.

Persistent homology measures the evolution of homology for the constitutive complexes in filtration. Once more and more cells are being added to a complex $\mathcal{C}_i$, new connected components or cycles may appear, old ones may become trivial or become identical (homologous) to others created earlier. For every connected component or a cycle,

[18] A number of points involved in the simplex minus one is a *dimension* of the simplex.

there are two important characteristics we will store: the first moment b, referred to as a *birth time*, when it appears in the filtration, and the last moment d, referred to as *death time*, when it either becomes trivial or becomes identical to other cycle created earlier. In this paper, instead of a standard birth-death summaries of persistent homology, we use birth-persistence coordinates, which can be obtained by the $[b, d] \to [b, d - b]$ transformation. The basic geometrical idea behind PH is presented in Fig. 11.

A couple of assumptions about PDs are made. Firstly, as our aim is to perform computations, we assume that persistence diagrams consist of finitely many points of nonzero persistence. Secondly, PDs may also contain infinite intervals that correspond to the so-called *essential classes*, i.e. the cycles that are born but never die. Those infinite intervals need to be processed prior to the computations. There are at least three strategies one can apply:

1. to ignore infinite intervals and use only the finite ones for consideration;
2. to substitute $+\infty$ in the death coordinates of the essential classes with a number N chosen by the user (a logical choice would be a number which is larger than a filtration value of any cell in the considered complex);
3. to build a pair of descriptors: one for finite, and one for infinite intervals and use them together as a final descriptor.

Given the available options, in the numerical experiments presented in this paper, we have chosen the simplest one, i.e. to ignore the infinite intervals. There are various classical metrics used to compare persistence diagrams (Edelsbrunner and Harer 2010). We will review them here, as they are essential in the study of stability of the presented representations. Note that the presentation is a bit non standard, as we are working on birth-peristence coordinates. Given two diagrams B and B', we construct a matching $\eta : B \to B'$ assuming that points can also be matched to $y = 0$ axis. Putting B and B' in the same diagram, one can visualize a matching η by drawing a line segment between $x \in B$ and $\eta(x)$ (note that $\eta(x)$ is either in B', or is a projection of x to its first coordinate). Given all the line segments, for each matching we can store the longest one, or a sum of lengths of all of them (raised to power q). Taking the minimum over all possible matchings of the obtained numbers will yield the *bottleneck* distance in the first case, and the *Wasserstein* distance (raised to power $\frac{1}{q}$) in the second case. More formally:

Definition *q-Wasserstein distance* between two persistence diagrams $B, B' \in \mathcal{D}$ is defined as:

$$W_q(B, B') := \left[\inf_{\eta : B \to B'} \sum_{x \in B} \|x - \eta(x)\|_\infty^q \right]^{\frac{1}{q}}.$$

In particular:

$$W_1(B, B') := \inf_{\eta : B \to B'} \sum_{x \in B} \|x - \eta(x)\|_\infty.$$

An important feature of persistent homology is its *stability*. Intuitively, it indicates that small changes in the filtration imply small changes (for instance in Wasserstein metric) in the resulting persistence diagrams. Formally:

Theorem *Edelsbrunner and Harer (2010) Let $\mathbb{X}$ be a finite cell complex and $f, g : \mathbb{X} \to \mathbb{R}$ filtering Lipshitz functions. Let B and B' be the PDs of $\mathbb{X}$ with filtration induced by f and g respectively. Then there exist constants C and k such that $W_1(B, B') \le C||f - g||_\infty^{1-k}$*

In this paper, we show stability with respect to 1-Wasserstein distance. Combined with the stability result described above, this indicates stability of bag-of-words representations with respect to the perturbation of initial data.

References

Aadcock ADRGC (2014) Classification of hepatic lesions using the matching metric. Comput Vis Image Underst 121:36–42

Adams H, Emerson T, Kirby M, Neville R, Peterson C, Shipman P, Chepushtanova S, Hanson E, Motta F, Ziegelmeier L (2017) Persistence images: a stable vector representation of persistent homology. J Mach Learn Res 18(8):1–35

Ali S, Basharat A, Shah M (2007) Chaotic invariants for human action recognition. In: ICCV, IEEE Computer Society, pp 1–8. http://dblp.uni-trier.de/db/conf/iccv/iccv2007.html#AliBS07

Anirudh R, Venkataraman V, Natesan Ramamurthy K, Turaga P (2016) A Riemannian framework for statistical analysis of topological persistence diagrams. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 68–76

Baeza-Yates R, Ribeiro-Neto B et al (1999) Modern information retrieval, vol 463. ACM press, New York

Bauer U, Kerber M, Reininghaus J, Wagner H (2017) Phat-persistent homology algorithms toolbox. J Symb Comput 78:76–90

Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517

Bubenik P (2015) Statistical topological data analysis using persistence landscapes. J Mach Learn Res 16(1):77–102

Cang Z, Wei G (2017) Topologynet: topology based deep convolutional and multi-task neural networks for biomolecular property predictions. PLoS Comput Biol 13(7):e1005690

Carrière M, Oudot S, Ovsjanikov M (2015) Stable topological signatures for points on 3d shapes. In: Computer graphics forum, vol 34, Wiley Online Library, pp 1–12

Carrière M, Cuturi M, Oudot S (2017) Sliced Wasserstein kernel for persistence diagrams. In: International conference on machine learning (ICML)

Chapelle O (2007) Training a support vector machine in the primal. Neural Comput 19(5):1155–1178

Chen C, Kerber M (2011) Persistent homology computation with a twist. In: Proceedings 27th European workshop on computational geometry, vol 11

Chen X, Golovinskiy A, Funkhouser T (2009) A benchmark for 3d mesh segmentation. In: ACM SIGGRAPH 2009 papers, SIGGRAPH '09, ACM, New York, NY, USA, pp 73:1–73:12. https://doi.org/10.1145/1576246.1531379

Cohen-Steiner D, Edelsbrunner H, Harer J (2007) Discrete Comput Geom. https://doi.org/10.1007/s00454-006-1276-5

DeGol J, Golparvar-Fard M, Hoiem D (2016) Geometry-informed material recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1554–1562

De Silva V, Morozov D, Vejdemo-Johansson M (2011) Dualities in persistent (co) homology. Inverse Probl 27(12):124003

Dey T, Shi D, Wang Y (2016) Simba: an efficient tool for approximating rips-filtration persistence via simplicial batch-collapse. In: Sankowski P, Zaroliagis C (eds) 24th annual European symposium on algorithms, ESA 2016, August 22–24, 2016, Aarhus, Denmark, LIPIcs, vol 57, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, pp 35:1–35:16. https://doi.org/10.4230/LIPIcs.ESA.2016.35

Di Fabio B, Ferri M (2015) Comparing persistence diagrams through complex vectors. In: International conference on image analysis and processing, Springer, pp 294–305

Donatini P, Frosini P, Lovato A (1998) Size functions for signature recognition. Proc SPIE 3454:178–183. https://doi.org/10.1117/12.323253

Edelsbrunner H, Letscher D, Zomorodian A (2002) Topological persistence and simplification. Discrete Comput Geom 28:511–533

Edelsbrunner H, Harer J (2010) Computational topology: an introduction. American Mathematical Soc

Ferri M (2017) Persistent topology for natural data analysis: a survey. In: Holzinger A, Goebel R, Palade V (eds) Towards integrative machine learning and knowledge extraction. Springer, Cham, pp 117–133

Ferri M, Frosini P, Lovato A, Zambelli C (1998) Point selection: A new comparison scheme for size functions (with an application to monogram recognition). In: Computer vision—ACCV'98: third Asian conference on computer vision Hong Kong, China, January 8–10, 1998 proceedings, vol I, Springer, pp 329–337

Gameiro M, Hiraoka Y, Izumi S, Kramar M, Mischaikow K, Nanda V (2015) A topological measurement of protein compressibility. Jpn J Ind Appl Math 32(1):1–17

Hofer C, Kwitt R, Niethammer M, Uhl A (2017) Deep learning with topological signatures. In: Advances in neural information processing systems, pp 1633–1643

Jégou H, Douze M, Schmid C, Pérez P (2010) Aggregating local descriptors into a compact image representation. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 3304–3311

Jégou H, Perronnin F, Douze M, Sánchez J, Perez P, Schmid C (2012) Aggregating local image descriptors into compact codes. IEEE Trans Pattern Anal Mach Intell 34(9):1704–1716

Kališnik S (2019) Tropical coordinates on the space of persistence barcodes. Found Comput Math 19(1):101–129. https://doi.org/10.1007/s10208-018-9379-y

Kerber M, Morozov D, Nigmetov A (2017) Geometry helps to compare persistence diagrams. J Exp Algorithm (JEA) 22:1–4

Kusano G, Fukumizu K, Hiraoka Y (2016) Persistence weighted Gaussian Kernel for topological data analysis. In: International conference on machine learning (ICML), vol 48

Lee H, Kang H, Chung M, Kim B, Lee D (2012) Persistent brain network homology from the perspective of dendrogram. IEEE Trans Med Imaging 31(12):2267–2277

Le T, Yamada M (2018) Persistence fisher kernel: a Riemannian manifold kernel for persistence diagrams. In: 32nd Conference on neural information processing systems (NeurIPS)

Li C, Ovsjanikov M, Chazal F (2014) Persistence-based structural recognition. In: IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 2003–2010. https://doi.org/10.1109/CVPR.2014.257

Liu J, Jeng S, Yang Y (2016) Applying topological persistence in convolutional neural network for music audio signals. arXiv preprint arXiv:1608.07373

Maria C, Boissonnat JD, Glisse M, Yvinec M (2014) The gudhi library: simplicial complexes and persistent homology. In: International congress on mathematical software, Springer, pp 167–174

McCallum A, Nigam K et al (1998) A comparison of event models for Naive Bayes text classification. In: AAAI-98 workshop on learning for text categorization, vol 752, pp 41–48

Monod A, Kališnik S, Patiño-Galindo JÁ, Crawford L (2019) Tropical sufficient statistics for persistent homology. SIAM J Appl Algebra Geom 3(2):337–371

Nakamura T, Hiraoka Y, Hirata A, Escolar EG, Nishiura Y (2015) Persistent homology and many-body atomic structure for medium-range order in the glass. Nanotechnology 26(30):304001

Nasrabadi NM (2007) Pattern recognition and machine learning. J Electron Imaging 16(4):049901

Perronnin F, Dance C (2007) Fisher kernels on visual vocabularies for image categorization. In: IEEE conference on computer vision and pattern recognition, 2007 CVPR'07, IEEE, pp 1–8

Perronnin F, Sénchez J, Xerox Y (2010) Large-scale image categorization with explicit data embedding. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 2297–2304

Poier G, Seidl M, Zeppelzauer M, Reinbacher C, Schaich M, Bellandi G, Marretta A, Bischof H (2017) Petrosurf3d: a dataset for high-resolution 3d surface segmentation. In: Proceedings of the 15th international workshop on content-based multimedia indexing (CBMI)

Reininghaus J, Huber S, Bauer U, Kwitt R (2015) A stable multi-scale kernel for topological machine learning. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 4741–4748. https://doi.org/10.1109/CVPR.2015.7299106

Seversky M, Davis S, Berger M (2016) On time-series topological data analysis: new data and opportunities. In: 2016 IEEE conference on computer vision and pattern recognition workshops (CVPRW), IEEE, pp 1014–1022

Sivic J, Zisserman A (2003) Video google: a text retrieval approach to object matching in videos. In: Ninth IEEE international conference on computer vision, IEEE, pp 1470–1477

Skraba P, Ovsjanikov M, Chazal F, Guibas L (2010) Persistence-based segmentation of deformable shapes. In: 2010 IEEE computer society conference on computer vision and pattern recognition workshops, pp 45–52. https://doi.org/10.1109/CVPRW.2010.5543285

Som A, Thopalli K, Karthikeyan NR, Vinay V, Shukla A, Pavan T (2018) Perturbation robust representations of topological persistence diagrams. In: European conference on computer vision, Springer, pp 638–659

Van Gemert JC, Geusebroek JM, Veenman CJ, Smeulders A (2008) Kernel codebooks for scene categorization. In: European conference on computer vision, Springer, pp 696–709

Vedaldi A, Fulkerson B (2008) VLFeat: an open and portable library of computer vision algorithms. http://www.vlfeat.org/

Vejdemo-Johansson M, Pokorny F, Skraba P, Kragic D (2015) Cohomological learning of periodic motion. Appl Algebra Eng Commun Comput 26(1):5–26. https://doi.org/10.1007/s00200-015-0251-x

Wang Z, Li Q, Li G, Xu G (2019) Polynomial representation for persistence diagram. In: Computer vision and pattern recognition (CVPR), IEEE, pp 6123–6132

Yanardag P, Vishwanathan S (2015) Deep graph kernels. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 1365–1374

Zeppelzauer M, Zieliński B, Juda M, Seidl M (2017) A study on topological descriptors for the analysis of 3d surface texture. Comput Vis Image Underst

Zieliński B, Lipiński M, Juda M, Zeppelzauer M, Dłotko P (2019) Persistence bag-of-words for topological data analysis. In: Proceedings of the international joint conference on artificial intelligence (IJCAI), Macao, China, pp 4489–4495. http://arxiv.org/abs/1802.04852

Zomorodian A, Carlsson G (2005) Computing persistent homology. Discrete Comput Geom 33(2):249–274