TDA-ML_Analysis

Project for the lecture: Geometric methods in data analysis*

Sebastian Dobrzynski, Daniel Gonzalez, André Schulze

October 30, 2019

Abstract

Topological Data Analysis (TDA) has shown to be a powerful tool that provides insight in the structure of data. Techniques of TDA, such as persistent homology, allow the study of qualitative features of data that persist across different predefined threshold values. The features can then be summarised in persistence diagrams for representation and further analysis. Nevertheless, as TDA is a recent field of study, there hasn't been given too much effort in comparing different standard Machine Learning (ML) methods with techniques in this field. In this work we want to overcome this issue, by comparing two well known ML approaches with two different multi-scale kernels for persistence diagrams. We analyse these methods using a handwritten data-set, as this has been studied thoroughly in the ML community.

1 Introduction

Topological Data Analysis (TDA) [2] is a recent field of study which makes use of different disciplines, such as algebraic topology, computational geometry and statistics [12] to get insights of the structure of data. This field has been increasingly studied over the last years, due to the well-understood theoretical framework, robustness over perturbations [15] and capability of extracting topological information from data. Persistent homology [9], the main TDA technique, allows to rank topological features according to a predefined filtration. Features that persist longer are ranked higher and the ones that are only visible at low levels are considered less relevant. The features can then be summarised in persistence diagrams [7] for representation and further analysis. Nevertheless, as TDA is a recent field of study, there hasn't been invested too much effort in comparing different standard Machine Learning methods with techniques in this field.

In this work we want to tackle this problem by analysing a bridge between Topology and Machine Learning (ML) provided by kernel-based learning methods derived from persistence diagrams. We also compare our results with two

*Lecture hold on the summer term of 2019 by Andreas Ott at the University of Heidelberg.

well known ML approaches, k-Nearest Neighbour (k-NN) and Convolutional Neural Networks (CNN). We follow [14] and [3] to define kernels for persistence diagrams, which prove to be stable with respect to the 1-Wasserstein distance. Using this approach, learning tasks such as classification can be performed directly with the advantage that they are generally efficient and easy to compute. We applied the above mentioned methods on the handwritten digits data-set [13], which contains 1797 hand-written digits of 8x8 pixels with their respective class labels.

The pipeline of this work will be presented as follows. In Section 2 we provide a concise and informal review of the theoretical background. We don't get into much details on the methods that we used, as they have been already presented in [3] and [14]. In Section 3 we present the approach that we used from the first to the last stage. For the computation of the persistent diagrams we used the tda-toolkit[1] using filtrations based on sublevel sets of the height function as in [11]. Our implementations of k-NN and CNN, is based on the sample code from a lecture on ML[2] and from an internet blog[3] respectively. Finally, the last section serves the evaluation and analysis of our results. Our implementations and data for the persistence diagrams can be found in https://github.com/DanielGonzalezAlv/TDA-ML_Analysis.

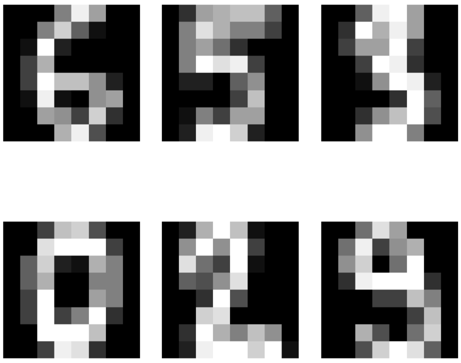

Figure 1: A few digits from the hand-written data-set

2 Theoretical Background

In this chapter we recall some theoretical background needed for our later computations. More specifically we review the kernels defined for persistence diagrams in [14] and [3], which we used in combination with a Support Vector

[1] https://github.com/c-hofer/tda-toolkit#references

[2] Fundamentals of Machine Learning hold on the winter term of 2018 at the University of Heidelberg

[3] https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/

Machine (SVM) algorithm. While persistent homology is the fundamental tool enabling us to compute persistence diagrams, one does not need to know the details for our discussion and therefore we refer to [8] for a general survey about this topic.

Persistence diagrams: The main output of persistent homology are the persistence diagrams. They are just multi-sets of points in $\mathbb{R}^2$, such that for every point (b, d), we have $d - b > 0$[4]. Here eb and d encode the first appearance (birth) and vanishing (death) of a topological feature. We denote the space of persistence diagrams by $\mathcal{D}$ and assume persistence diagrams to always be finite, since this is always the case when working with a finite set of data. For each $n \in \mathbb{N} \cup \{\infty\}$ we have a metric d_n which we call the *n-Wasserstein metric.* One has

$$\lim_{n \in \mathbb{N}} d_n = d_\infty$$

For a definition of the d_n metric, we refer to [8]. Unfortunately the space $\mathcal{D}$ does not admit a Hilbert space structure with respect to the Wasserstein distances, though persistence diagrams can't be used in standard ML methods such as SVM and Principal Component Analysis (PCA) in a straightforward manner, also the metrics d_n are not conditionally negative semi-definite which makes non trivial the task of deriving kernels for persistence diagrams. Nevertheless, a lot of progress has been made in this area.

There are essentially two classes of proposed Kernels for persistence diagrams. The first class of methods builds explicit feature maps. A known example for this class are Persistence Landscapes [1]. The second class defines feature maps implicitly and focuses on building a kernel for persistence diagrams. In this work we used kernels suggested in [14] and [3], which belong to the latter class and combine them with a SVM algorithm.

Kernels: Let us quickly review what a kernel is. Given a set X, a function $k : X \times X \longrightarrow \mathbb{R}$ is called a (positive definite) kernel if for all $n \in \mathbb{N}$ and all families of points $x_1, \dots, x_n$ in X, the Gramian matrix $[k(x_i, x_j)]_{i,j}$ is positive semi definite. In that case there exists a so called feature map $\phi : X \longrightarrow H$ into a Hilbert space H, such that $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_H$ and we can define a distance on X:

$$d_k(x_1, x_2) \coloneqq (k(x_1, x_1) + k(x_2, x_2) - k(x_1, x_2))^{\frac{1}{2}}.$$

A common way to define a kernel is to set

$$k(x_1, x_2) \coloneqq \exp\left(-\frac{f(x_1, x_2)}{2\sigma}\right)$$

where f is a conditionally negative definite function and $\sigma > 0$. In practice the scale factor σ is adjusted to best fit the data. Of course k depends on the value of sigma, but we will leave it out of the notation for simplicity.

Sliced Wasserstein kernel: The first kernel that we implemented, is the so called *sliced Wasserstein Kernel*, which is defined in [3]. In this section we summarise the results from [3] briefly.

[4] In general we can have $d = \infty$. That means that the feature never vanishes in the filtration. (b, d) is then called an essential feature.

Definition 1. Let μ and ν be two non negative measures on the real line such that $\mu(\mathbb{R}) = \nu(\mathbb{R}) \coloneqq r$. We define the 1-*Wasserstein Kernel* as

$$\mathcal{W}(\mu, \nu) \coloneqq \inf_{P \in \Pi(\mu,\nu)} \int\limits_{\mathbb{R}\times\mathbb{R}} \mid x - y \mid P(dx, dy),$$

where $\Pi(\mu, \nu)$ is the set of measures on $\mathbb{R}^2$ with marginals μ and ν.

The essential result is

Proposition 1. $\mathcal{W}$ is conditionally negative definite on the space of measures with mass r.

Proof. see [3], Proposition 2.1 □

Remark. Let $\mu = \sum_{i=1}^{n} \delta_{x_i}$ and $\nu = \sum_{i=0}^{n} \delta_{y_i}$ where δ_{x_i} and δ_{y_i} are Dirac measures with ordered $x_1 \leq \cdots \leq x_n$ and $y_1 \leq \cdots \leq y_n$, i.e. μ and ν are empirical measures. In this situation $\mathcal{W}$ reduces to

$$\mathcal{W}(\mu, \nu) = \sum_i \mid x_i - y_i \mid = \parallel x - y \parallel_1 .$$

To a persistence diagram Dg_i we can assign empirical measure in the following way. Let θ be an element in the unit Sphere $S^1 \subseteq \mathbb{R}^2$ and Let $L(\theta)$ denote the 1-dimenional linear space generated by θ. Furthermore let π_θ(resp.π_Δ) denote the orthogonal projection from $\mathbb{R}^2$ into $L(\theta)$(resp.the diagonal). We then define measures $\mu_i^\theta \coloneqq \sum_{p \in Dg_i} \delta_{\pi_\theta(p)}$ and $\mu_{i\Delta}^\theta \coloneqq \sum_{p \in Dg_i} \delta_{\pi_\theta \circ \pi_\Delta(p)}$.

Definition 2. Let Dg_1 and Dg_2 be two persistence diagrams the *sliced Wasserstein metric* is defined as

$$SW(Dg_1, Dg_2) \coloneqq \frac{1}{2\pi} \int_{S^1} \mathcal{W}(\mu_1^\theta + \mu_{1\Delta}^\theta, \mu_2^\theta + \mu_{2\Delta}^\theta) d\theta.$$

Since $\mathcal{W}$ is conditionally negative definite SW itself is. we can therefore define a valid kernel by setting

$$k_{sw}(Dg_1, Dg_2) \coloneqq \exp\left(-\frac{SW(Dg_1, Dg_2)}{2\sigma^2}\right).$$

Now in [3], Theorems 3.3 and 3.4 it the Authors proof that SW is stable and discriminative with respect to the first 1-Wasserstein metric d_1, that is, there exists positive numbers M_1 and M_2 such that

$$SW(Dg_1, Dg_2) \leq M_1 d_1(Dg_1, Dg_2) \text{ and } M_2 d_1(Dg_1, Dg_2) \leq SW(Dg_1, Dg_2),$$

for all persistence diagrams Dg_1 and Dg_2 (Note that M_2 depends on the number of points in the diagrams). Therefore it is reasonable to assume that k_{sw} will do a good job in classifying persistence diagrams.

In practice one can approximate k_{sw} in $O(MN\log(N))$ time. Where N is equal to $\max\{\mid Dg_1 \mid, \mid Dg_2 \mid\}$ and M is the number of directions we choose in S^1. This works because the authors of [3] show empirically that just a few directions are sufficient to get good classification accuracies(see section 4). You can find our implementation of the code at the end of this work.

Persistence scale-space kernel: Let us now summarise the results from [14]. The authors use the heat diffusion partial differential equation to build a feature map

$$\phi : \mathcal{D} \longrightarrow L_2(\Omega), \text{ where } \Omega = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 \geq x_1\}.$$

One then gets a Kernel by setting

$$k_{ss}(Dg_1, Dg_2) \coloneqq \langle \phi(Dg_1), \phi(Dg_2) \rangle_{L_2(\Omega)}.$$

Note that ϕ and k_{ss} again depend on a scale parameter $\sigma > 0$, but for simplicity we leave σ out of the notation. The distance induced by the k_{ss} is just the usual distance on $L_2(\Omega)$. The Authors then derive a closed expression to calculate k_{ss}. Namely,

$$k_{ss}(Dg_1, Dg_2) = \frac{1}{8\pi\sigma} \sum_{\substack{p \in Dg_1, \\ q \in Dg_2}} \exp\left(-\frac{\| p - q \|^2}{8\sigma}\right) - \exp\left(\frac{\| p - \overline{q} \|^2}{8\sigma}\right).$$

k_{ss} can be calculated in $O(|Dg_1| \cdot |Dg_2|)$ time or in $O(|Dg_1| + |Dg_2|)$ time with a bounded error. The usefulness of this Kernel comes from

Proposition 2. The kernel k_{ss} is stable with respect to d_1, more precisely we have

$$||\phi(Dg_1) - \phi(Dg_2)||_{L_2(\Omega)} \leq \frac{1}{2\sqrt{\pi}\sigma} d_1(Dg_1, Dg_2)$$

Proof. see [14], Theorem 2. □

Remark. One can show that a non trivial kernel can not be stable with respect to d_p for $p \neq 1$(see [14], Theorem 3.).

k-NN and CNN: We will later compare our topological approach with two of the most commonly used classification algorithms. Introducing them thoroughly would go way beyond the scope of this work. Therefore we only say a few words for each and recommend some literature for further reading.

- The k-NN approach is a non-parametric classification method that can also be used for regression. A point is assigned to the class that is most common among it's k nearest neighbours with respect to the appropriate distance, where the best choice of $k > 1$ depends on the given data. There are also weighted versions of k-NN but we won't use this in our work. Since there are many textbooks that cover this topic, we won't give further details

- Convolutional Neural Networks (CNN) are inspired, but still rather different to biological neural networks. A CNN typically consists of an input layer, an output layer and multiple so called hidden layers. One of the major strengths of the use of neural networks in machine learning is there need of relatively little pre-processing.

3 Experiments

The usefulness of the kernel functions arise from the Representation theorem. It allows to reduce an empirical risk minimisation problem, from an infinite dimensional space, to a finite dimensional optimisation problem. More formally, a minimizer f^* of a regularised empirical risk functional, derived from a reproducing kernel Hilbert space, might be represented as a finite linear combination of kernel products evaluated on input points of the training data. This is exploited by a class of algorithms called kernel methods. Instead of explicitly computing data points, they operate on the feature space using the inner products, which is usually cheaper. Some representatives include SVM, kernel Perceptron and Gaussian Processes.

In our experiments, we implement the Wasserstein kernel (SW-Kernel) [3] and the scale-space kernel (MS-Kernel) [14] by first computing the Gram matrix for each of them. We tune empirically the parameters for both kernels, using ... Note that this has complexity $\mathcal{O}(n^2)$, which is the limiting factor in training. Then the Gram matrix is fed into the SVM of the python sklearn module, which is based on the reference implementation of libsvm [4]. To predict values all inner products of the training and test set have to be supplied. Clearly this has undesired complexity for testing, but is a limitation of the python module and not the approach itself. As a result, no cross-validation was used in the experiments, which would be a standard measure otherwise. Finally, for evaluating and analysing our results, we solve the classification problem computing the k-NN algorithm (with $k = 5$ and $k = 100$) and a CNN.

Using these kernel methods is only meaningful, if the persistence diagrams are able to encode the image information. Therefore, it is necessary to choose an appropriate filtration for the given data-set. We choose the tda-toolkit mentioned in the Introduction with 4 and 8 angle directions for the filtration. The persistence diagrams are computed with the 2 dimensional persistent homology transform (PHT) [10], which also employs the DIPHA package [6]. The PHT is a special method to define a filtration through the mapping $x \mapsto \langle x|v \rangle$, $x \in \mathbb{R}^n$ for each $v \in \mathcal{S}^{n-1}$, as first described by [16]. Notice that this algorithm returns one persistence diagram for each angle $v \in \mathcal{S}^{n-1}$ and homology group. Therefore we define the inner product of a set of persistence diagrams as the sum of inner products of the same angle and homology group.

4 Results and conclusion

We compare the results of the methods that we used by computing the accuracy with respect to the training-set size. We fixed the test-data, by choosing one-tenth of the hole data-set randomly. Then, we compute the accuracy by training our algorithms increasingly with the rest of the data. Figure 2 illustrates the outcome of our experiments.

One can see that the SW-kernel performs slightly better than the MS-kernel. This might comes from the fact that the SM-kernel is not only stable but also disciminative with respect to the 1-Wasserstein distance. Unsurprisingly the CNN and k-NN algorithm outperformed the persistence kernel-methods. Neural Networks are known to achieve the highest accuracy on the MNIST data set [5], which is comparable to the hand-written data-set that we used. We think that

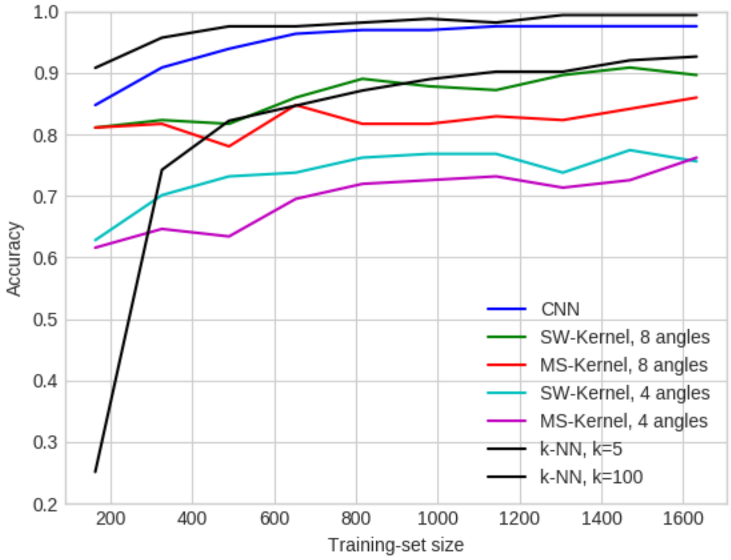

Figure 2: Accuracy with respect to the training-set size.
Parameters for the SW-Kernel: $m = 5$, $\sigma = 0.5$; for the MS-Kernel: $\sigma = 0.001$ and for the persistent homology transform $n = 8$.

the well performance of the k-NN approach is due to the fact that hand-written images are in low resolution. They are therefore centered and suitable for pixel-wise comparison. The kernel-based methods using 8 angle directions achieve an accuracy over 80 per cent, while using 4 angles don't accomplish this results. We could probably improve this results, using a better systematic approach to determine the parameters σ and m.

Besides the lower performance of the kernel-based approaches, we observe that TDA methods can be used as a ML approach with high accuracy. Considering the fact that the pictures in our data-set are in low resolution, and the digits don't differ to drastically in their topological features, the outcome seems fairly remarkable, and we can expect even better outcomes with more complex data. We think that improving the resolution of the data may improve our results. As a final remark, we tried to improve the quality of the data-set by applying the Otsu's thresholding method on it. The results were slightly worse and therefore we didn't present it here. For more details about the whole implementation and data that we compute, we refer to https://github.com/DanielGonzalezAlv/TDA-ML_Analysis.

References

[1] Peter Bubenik and Paweł Dłotko. "A persistence landscapes toolbox for topological statistics". In: *Journal of Symbolic Computation* 78 (Jan. 2017),

pp. 91–114. ISSN: 0747-7171. DOI: 10.1016/j.jsc.2016.03.009. URL: http://dx.doi.org/10.1016/j.jsc.2016.03.009.

[2] Gunnar Carlsson. "Topology and data". In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308.

[3] Mathieu Carrière, Marco Cuturi, and Steve Oudot. *Sliced Wasserstein Kernel for Persistence Diagrams.* 2017. arXiv: 1706.03358 [cs.CG].

[4] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

[5] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *IN COMPUTER VISION AND PATTERN RECOGNITION.* 2012, pp. 3642–3649.

[6] *DIPHA A Distributed Persistent Homology Algorithm.* https://github.com/DIPHA/dipha. Accessed: 20109-10-20.

[7] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction.* American Mathematical Soc., 2010.

[8] Herbert Edelsbrunner and John Harer. "Persistent homology—a survey". In: *Discrete Computational Geometry - DCG* 453 (Jan. 2008). DOI: 10.1090/conm/453/08802.

[9] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. "Topological persistence and simplification". In: *Proceedings 41st Annual Symposium on Foundations of Computer Science.* IEEE. 2000, pp. 454–463.

[10] Christoph Hofer et al. "Constructing Shape Spaces from a Topological Perspective". In: *Information Processing in Medical Imaging.* Cham: Springer International Publishing, 2017, pp. 106–118. ISBN: 978-3-319-59050-9.

[11] Christoph Hofer et al. "Deep learning with topological signatures". In: *Advances in Neural Information Processing Systems.* 2017, pp. 1634–1644.

[12] Nina Otter et al. "A roadmap for the computation of persistent homology". In: *EPJ Data Science* 6.1 (2017), p. 17.

[13] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] Jan Reininghaus et al. *A Stable Multi-Scale Kernel for Topological Machine Learning.* 2014. arXiv: 1412.6821 [stat.ML].

[15] Anirudh Som et al. "Perturbation robust representations of topological persistence diagrams". In: *Proceedings of the European Conference on Computer Vision (ECCV).* 2018, pp. 617–635.

[16] Katharine Turner, Sayan Mukherjee, and Doug M Boyer. *Persistent Homology Transform for Modeling Shapes and Surfaces.* 2013. arXiv: 1310.1030 [math.ST].