DEGREE PROJECT IN MATHEMATICS,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Statistical Learning and Analysis on Homology-Based Features

JENS AGERBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Statistical Learning and Analysis on Homology-Based Features

JENS AGERBERG

Degree Projects in Mathematical Statistics (30 ECTS credits)
Master's Programme in Applied and Computational Mathematics
KTH Royal Institute of Technology year 2020
Supervisor at RISE SICS: Ather Gattami
Supervisors at KTH: Martina Scolamiero, Wojciech Chachólski
Examiner at KTH: Martina Scolamiero

TRITA-SCI-GRU 2020:056
MAT-E 2020:019

Royal Institute of Technology
School of Engineering Sciences
KTH SCI
SE-100 44 Stockholm, Sweden
URL: www.kth.se/sci

Abstract

Stable rank has recently been proposed as an invariant to encode the result of persistent homology, a method used in topological data analysis. In this thesis we develop methods for statistical analysis as well as machine learning methods based on stable rank. As stable rank may be viewed as a mapping to a Hilbert space, a kernel can be constructed from the inner product in this space. First, we investigate this kernel in the context of kernel learning methods such as support-vector machines. Next, using the theory of kernel embedding of probability distributions, we give a statistical treatment of the kernel by showing some of its properties and develop a two-sample hypothesis test based on the kernel. As an alternative approach, a mapping to $\mathbb{R}^d$ with learnable parameters can be conceived, serving as an input layer to a neural network. The developed methods are first evaluated on synthetic data. Then the two-sample hypothesis test is applied on the OASIS open access brain imaging dataset. Finally a graph classification task is performed on a dataset collected from Reddit.

Statistisk analys och maskininlärning med homologibaserad data

Sammanfattning

Stable rank har föreslagits som en sammanfattning på datanivå av resultatet av *persistent homology*, en metod inom topologisk dataanalys. I detta examensarbete utvecklar vi metoder inom statistisk analys och maskininlärning baserade på stable rank. Eftersom stable rank kan ses som en avbildning i ett Hilbertrum kan en kärna konstrueras från inre produkten i detta rum. Först undersöker vi denna kärnas egenskaper när den används inom ramen för maskininlärningsmetoder som stödvektormaskin (SVM). Därefter, med grund i teorin för inbäddning av sannolikhetsfördelningar i *reproducing kernel* Hilbertrum, undersöker vi hur kärnan kan användas för att utveckla ett test för statistisk hypotesprövning. Slutligen, som ett alternativ till metoder baserade på kärnor, utvecklas en avbildning i $\mathbb{R}^d$ med optimerbara parametrar, som kan användas som ett ingångslager i ett neuralt nätverk. Metoderna utvärderas först på syntetisk data. Vidare utförs ett statistiskt test på OASIS, ett öppet dataset inom neuroradiologi. Slutligen utvärderas metoderna på klassificering av grafer, baserat på ett dataset insamlat från Reddit.

Acknowledgements

I would like to thank Martina Scolamiero and Wojciech Chachólski at KTH for introducing me to the field of topological data analysis and being so generous with their time, support and engagement in this work.

I would also like to thank Ather Gattami at RISE for valuable feedback and guidance throughout the thesis.

Contents

Chapter 1

Introduction

1.1 Background

Topological data analysis (TDA) constitutes a novel framework for data analysis, mathematically well-founded and with roots in algebraic topology. Through the method of persistent homology, TDA proposes to analyze datasets, often high-dimensional and unstructured, but where each observation is a space that encodes some notion of distance. An example of such space is a point cloud, which is produced for instance by 3D scanners. Persistent homology represents such spaces through a combinatorial object, a parametrized simplicial complex, from which topological features can be computed at various spatial resolutions. Such topological summaries, which can be seen as the output of persistent homology, are robust to perturbations in the input data and their efficient computation has been made possible by readily available tools. In recent applications, and in such varied fields as bioinformatics [36] and finance [22], it has been shown to encode valuable information, often orthogonal to that derived from non-topological methods.

To take the data analysis further, one would like to introduce a statistical framework, for instance to be able to consider probability distributions over topological summaries and infer their properties based on samples. Alternatively, the discriminative information contained in the topological summaries makes them interesting in the context of machine learning, for instance to serve as input in supervised learning problems. The space of topological summaries – called persistence diagrams – while endowed with a metric, suffers from computational challenges and lacks the structure

of a Euclidean or more generally Hilbert space often desired for the development of machine learning methods. For these reasons, much of the recent efforts to develop statistical methods for topological summaries have been devoted to crafting mappings from the space of persistence diagrams to spaces where statistical concepts are well-founded. Such mappings need to be stable, i.e. robust to perturbations in the input, computationally efficient and retain the discriminative information of the summaries.

Instead of considering mappings from the space of persistence diagrams, Chachólski and Riihimäki [11] propose to work in the space of parametrized vector spaces, obtained during the computation of persistent homology. This space can be endowed with various metrics and possesses a discrete invariant, called rank. The authors propose a hierarchical stabilization of the rank leading to a new invariant, *stable rank*, with a richer geometry. By construction, the stable rank encodes the metric, chosen to compare parametrized vector spaces. Except for a proof of concept supplied in [11], stable rank has yet to be adapted to a statistical framework or be used for machine learning problems. The approach is however believed to present advantages. Indeed, instead of a complete representation of persistence, as in the case of persistence diagrams, stable rank allows to choose a suitable metric to highlight properties of a particular dataset and problem at hand. For instance for a classification problem, a metric that discriminates well between classes could be chosen. As stable rank constitutes a mapping into L^p spaces, many statistical methods can be envisioned.

1.2 Objective

Our objective in this thesis is to develop methods in statistical analysis and learning for persistent homology, based on stable rank. More specifically:

1. We first construct kernels based on stable rank: as stable rank may be viewed as a mapping to a Hilbert space, a kernel can be constructed from the inner product in this space. Such a kernel can be used in the context of kernel learning methods such as support-vector machines (SVM).

2. Such kernels can also be used for statistical analysis, motivated by the theory of kernel embedding of probability distributions. We will show how this embedding

justifies the construction of hypothesis tests based on the kernel, which can be used to determine if two samples of outputs of persistent homology can be said to originate from the same probability distribution.

3. Finally, as an alternative to kernel approaches, we conceive a mapping with learnable parameters to $\mathbb{R}^d$, serving as an input layer to a neural network. Optimizing the parameters of such mapping can be seen as a way to find an optimal metric for a problem at hand, within the framework of stable rank.

1.3 Organization

In chapter 2 the relevant background and related work is presented. In chapter 3 the kernels based on stable rank are described, and their use in the context of machine learning methods and kernel-based statistical methods is motivated. In chapter 4 the input layer to a neural network, derived from stable rank, is described. Chapter 5 contains the experiments. First, the statistical tests are performed on synthetic data and on the OASIS open access brain imaging dataset. Second, the machine learning methods are evaluated on synthetic data and on a graph classification task based on a dataset collected from Reddit. Finally chapter 6 summarizes the thesis and discusses future directions.

Chapter 2

Background and related work

2.1 Persistent homology

2.1.1 Point clouds and distance spaces

The most fundamental way of encoding spatial information is via distances. A distance on a set X is by definition a function $d\colon X \times X \to \mathbb{R}$ such that it is non-negative $d(x, y) \geq 0$, symmetric $d(x, y) = d(y, x)$ and reflexive $d(x, x) = 0$ for all x and y in X. If in addition triangle inequality holds, i.e. $d(x, y) + d(y, z) \geq d(x, z)$, for all x, y, z in X, then such a distance d is called a metric. For example the function that assigns to two elements x and y in $\mathbb{R}^n$ the number $\sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$ is the well-known Euclidean metric. This is only one of many possible distances on $\mathbb{R}^n$.

A point cloud is by definition a finite subset of $\mathbb{R}^n$ with the distance given by the restriction of a chosen distance on $\mathbb{R}^n$. More generally, distance spaces can encode a notion of distance (or similarity) between a finite set of points, e.g. in a pairwise distance matrix.

Such structures are often outcomes of experiments and measurements done during experiments. They are the main class of objects that topological data analysis aims at understanding through the method of the persistent homology. In this thesis the main running theme is to study statistical behaviour of homology-based invariants extracted from distance spaces.

Although distances are convenient in encoding outcomes of experiments, they are not convenient in extracting homology of the representing spatial information. For that

purpose simplicial complexes will be used. Moreover, some data, such as graphs or polygon meshes, can be directly modeled as simplicial complexes, as is demonstrated in some of the experiments in chapter 5.

2.1.2 Simplicial complexes

A collection K of finite and non-empty subsets of X is called a simplicial complex (on X) if:

- $\{x\}$ belongs to K for any x in X.
- if σ belongs to K, then so does any non empty subset of σ.

An element $\sigma = \{x_0, \ldots, x_n\}$ in a simplicial complex K is called a simplex of dimension n.

The subset of K consisting of simplices of dimension n is denoted by K_n. Elements of K_0 are referred to as vertices and elements of K_1 as edges.

For example the collection $\Delta[X]$, of all finite subsets of X, is a simplicial complex.

Let K be a simplicial complex on X and L be a simplicial complex on Y. Then a function $f\colon K \to L$ is called a map of simplicial complexes if for every σ in K there is a function $f_0\colon X \to Y$ such that $f(\sigma) = \{f_0(x) \text{ s.t. } x \text{ in } \sigma\}$ belongs to L.

Simplicial complexes are combinatorial objects which encode spatial information in a way which is convenient for extracting homology. An example simplicial complex is shown in Figure 2.1.1.

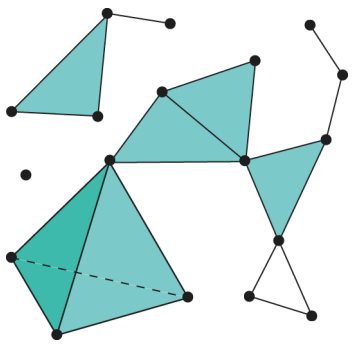


Figure 2.1.1: Example of a simplicial complex.

2.1.3 Homology on simplicial complexes

Our first step in defining homology is to move from simplicial complexes to vector spaces. In this thesis we are going to consider only homology with coefficients in the field $\mathbf{F}_2$ of two elements. Let K be a simplicial complex and n a natural number. We define $\mathbf{F}_2 K_n$ to be the $\mathbf{F}_2$-vector space with K_n as its base. Explicitly:

$$\mathbf{F}_2 K_n = \oplus_{\sigma \in K_n} \mathbf{F}_2. \tag{2.1}$$

Our next step is to define, for $n > 0$, a linear function on bases elements:

$$\begin{aligned} &\partial_n : \mathbf{F}_2 K_n \to \mathbf{F}_2 K_{n-1} \\ &\partial_n(\sigma) := \sum_{\tau \subset \sigma, \tau \in K_{n-1}} \tau. \end{aligned} \tag{2.2}$$

In addition, set ∂_0 to be the zero function $\partial_0 = 0 \colon \mathbf{F}_2 K_0 \to 0$. The composition $\partial_n \circ \partial_{n+1}$ is the zero function for all natural numbers n. This implies that $\mathrm{Im}(\partial_{n+1}) \subset \mathrm{Ker}(\partial_n)$. Elements in $\mathrm{Ker}(\partial_n)$ are called cycles and elements in $\mathrm{Im}(\partial_{n+1})$ are called boundaries.

We can thus define the following quotient vector space, which is called the n-th homology of K:

$$H_n(K) := \mathrm{Ker}(\partial_n)/\mathrm{Im}(\partial_{n+1}). \tag{2.3}$$

Homology measures how many cycles which are not also boundaries there are. The dimension of $H_n(K)$, $\dim(\mathrm{Ker}(\partial_n)) - \dim(\mathrm{Im}(\partial_{n+1}))$, called the n-th Betti number, is of particular importance as it can be interpreted as capturing the number of topological features of the simplicial complex, e.g. number of connected components for $H_0(K)$, number of holes for $H_1(K)$ and similarly for $n > 1$.

Let $f \colon K \to L$ be a map of simplicial complexes. For every natural number n define a

linear function $f_n \colon \mathbf{F}_2 K_n \to \mathbf{F}_2 L_n$ on bases elements as follows:

$$f_n(\sigma) := \begin{cases} f(\sigma) & \text{if } f(\sigma) \text{ is a simplex of dimension } n, \\ 0 & \text{if } f(\sigma) \text{ is a simplex of dimension strictly smaller than } n. \end{cases}$$

The sequence consisting of the linear functions $f_n \colon \mathbf{F}_2 K_n \to \mathbf{F}_2 L_n$ indexed by the natural numbers makes the following diagram commutative for every positive n:

$$\begin{array}{ccc} \mathbf{F}_2 K_{n+1} & \xrightarrow{\partial_{n+1}} & \mathbf{F}_2 K_n \\ {\scriptstyle f_{n+1}}\downarrow & & \downarrow{\scriptstyle f_n} \\ \mathbf{F}_2 L_{n+1} & \xrightarrow{\partial_{n+1}} & \mathbf{F}_2 L_n \end{array}$$

Hence f_n maps boundaries and cycles of K into respectively boundaries and cycles of L, and thus induces a linear map on the homology. We denote this linear function as:

$$H_n f \colon H_n K \to H_n L.$$

The assignment $f \mapsto H_n f$ satisfies the following properties: it preserves the identity $H_n(\mathrm{id}) = \mathrm{id}$ and it commutes with compositions $H_n(f \circ g) = (H_n f) \circ (H_n g)$ for any composable maps f and g of simplicial complexes. This means that homology is a functor transforming simplicial complexes and their maps into vector spaces with linear functions as morphisms.

2.1.4 Vietoris-Rips

We have so far discussed homology of simplicial complexes. While this can be interesting in its own right, the goal as introduced in 2.1.1 is often to study homology (and further homology-based invariants and its statistical properties) of distance spaces. In doing so we need a translation step that transforms distance information into a simplicial complex. In this thesis we focus on Vietoris-Rips construction for this purpose.

Let d be a distance on a finite set X. Choose a non-negative real number t in $[0, \infty)$. Then the collection:

$$VR_t(X,d) := \{\sigma \subset X \text{ s.t. } \sigma \text{ is finite and } d(x,y) \leq t \text{ for all } x,y \in \sigma\} \tag{2.4}$$

is a simplicial complex called the Vietoris-Rips complex of the distance d at time t. In words, all subsets of X where all elements are at distance at most t of each other will form simplices. Note that if $s < t$, then there is an inclusion $\text{VR}_s(X,d) \subset \text{VR}_t(X,d)$ which is a map of simplicial complexes. Note further that since X is finite, there is t such that $\text{VR}_t(X,d) = \Delta[X]$. Vietoris-Rips at time t is illustrated in Figure 2.1.2.

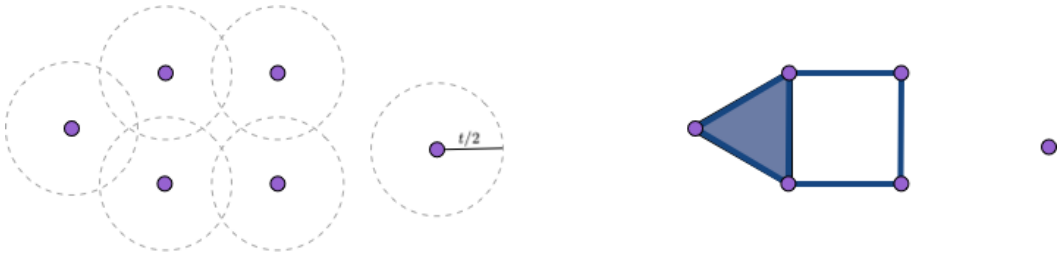


Figure 2.1.2: **Left**: six purple points together with balls of radius $\frac{t}{2}$ allowing to judge the distance between the points. **Right**: the Vietoris-Rips complex at time t. The points form simplices of dimension 0, the blue lines correspond to simplices of dimension 1 and the filled triangle corresponds to a simplex of dimension 2.

2.1.5 Parametrized simplicial complexes and tameness

Let d be a distance on a set X. The family of maps between Vietoris-Rips complexes of a given distance $\{\text{VR}_s(X,d) \subset \text{VR}_t(X,d)\}_{s<t\in[0,\infty)}$ is an example of a $[0,\infty)$-parametrized simplicial complex, which we denote simply by $\text{VR}(X,d)$. A $[0,\infty)$-parametrized simplicial complex K is by definition a functor indexed by the poset category $[0,\infty)$, of non negative real numbers, with values in the category of simplicial complexes. Explicitly, a $[0,\infty)$-parametrized simplicial complex is a sequence of simplicial complexes $\{K_s\}_{s\in[0,\infty)}$, indexed by $[0,\infty)$, together with a sequence of maps $\{K_{s<t}\colon K_s \to K_t\}_{s<t\in[0,\infty)}$, indexed by $s < t$ in $[0,\infty)$, such that $K_{s<t}K_{r<s} = K_{r<t}$, for every $r < s < t$ in $[0,\infty)$.

By taking homologies we can transform a $[0,\infty)$-parametrized simplicial complex into a $[0,\infty)$-parametrized vector space V. Explicitly a parametrized vector space is a collection of vector spaces $\{V_s\}_{s\in[0,\infty)}$ together with maps between vector spaces $\{V_{s<t} : V_s \to V_t\}_{s<t\in[0,\infty)}$ for $s < t$ in $[0,\infty)$ such that $V_{s<t}V_{r<s} = V_{r<t}$, for every $r < s < t$ in $[0,\infty)$.

If X is a finite set, then this parametrized vector space satisfies additional tameness conditions:

1. V_s is finite-dimensional for any scale s in $[0, \infty)$.
2. There is a finite sequence $0 < t_0 < ... < t_k$ of elements in $[0, \infty)$ such that for $s \leq t$ in $[0, \infty)$, a transition map $V_{s \leq t}$ may fail to be an isomorphism only if $s < t_i \leq t$ for some i. Such a sequence $t_0, ..., t_k$ is called a discretizing sequence of size $k+1$.

Note that a sequence $t_0, ..., t_k$ discretizes a tame parametrized vector space V, if and only if the restrictions of V to the subposets $[0, t_0)$, ..., $[t_{k-1}, t_k)$, $[t_k, \infty)$ are isomorphic to the constant functors.

We denote the set of tame parametrized vector spaces as $\mathrm{Tame}([0, \infty), \mathrm{Vect})$.

Proposition 1. *Let V be a tame parametrized vector space. Then there is a unique minimal (with respect to the size) discretizing sequence for V.*

Proof. We can assume that the parametrized vector space V is not constant, otherwise $t = \{\emptyset\}$ is a discretizing sequence of size zero and therefore the unique minimal discretizing sequence for V. The set X of all sequences discretizing V is not empty and its elements have size at least one, since V is tame and non-constant. Let $\tau = \{\tau_i\}_{i=0}^{k}$ be an element in X of minimal size, i.e. $|\tau| \leq |x|, \forall x \in X$. By minimality of τ and the fact that it is a discretizing sequence, for each $\tau_i \in \tau$ there exists $(a_i, b_i) \in [\tau_{i-1}, \tau_{i+1})$ such that $V_{a_i < b_i}$ is not an isomorphism and $a_i < \tau_i < b_i$. It follows that $V_{s<t}$ is also not an isomorphism for $a_i \leq s < \tau_i < t \leq b_i$.

We will prove that $\tau \subset x, \forall x \in X$. Thus this discretizing sequence can be characterized as $\tau = \bigcap_{x \in X} x$ and is therefore unique. Suppose there exists $\tau^{'}$ such that τ is not a subset of $\tau^{'}$. In particular there exists $\tau_i \in \tau$ such that $\tau_i \neq \tau_j^{'}$ for all $\tau_j^{'} \in \tau^{'}$. Define $\epsilon = \min_j |\tau_i - \tau_j^{'}|$. Then for $a = \max\{a_i, \tau_i - \frac{\epsilon}{2}\}$ and $b = \min\{b_i, \tau_i + \frac{\epsilon}{2}\}$, we have that $V_{a<b}$ is not an isomorphism although there is no $\tau_j^{'}$ in the interval $[a, b)$, contradicting the hypothesis that $\tau^{'}$ is a discretizing sequence for V. □

2.1.6 Bar decomposition

Our goal is now to move towards a representation of the n-th persistent homology useful for data analysis. We start by defining the parametrized vector space $K(s, e)$:

$$
\begin{aligned}
K(s,e)_a &:= \begin{cases} K & \text{if } s \leq a < e, \\ 0 & \text{otherwise,} \end{cases} \\
K(s,e)_{a \leq b} &:= \begin{cases} \text{id} & \text{if } \dim(K(s,e)_a) = \dim(K(s,e)_b) = 1, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{2.5}
$$

These are indecomposable tame parametrized vector spaces and hence constitute the simplest possible objects of this class. One of the fundamental theorems in Persistent homology [55] shows that any $V \in \text{Tame}([0,\infty), \text{Vect})$ is isomorphic to a direct sum $\bigoplus_{i=1}^{n} K(s_i, e_i)$. Such collections of bars are called barcodes and can be represented as multisets of elements in the upper half-plane, that is a collection of points $\{(s_i, e_i)$ s.t. $(s_i, e_i) \in \mathbb{R}^2, e_i > s_i\}_{i=1}^{n}$ for some n where all elements (s_i, e_i) need not be unique. Such multisets constitute a form of topological summary called persistence diagrams and denoted by $\mathcal{D}$. From a persistence diagram plotted in the plane, one can read off when (in the sense of the filtration scale, e.g. t for Vietoris-Rips above) "topological features" appeared (s_i) and how long they persisted (i.e. $e_i - s_i$ if $e_i < \infty$, if $e_i = \infty$ the topological feature persists forever and is called essential), as well as how many such features there were (the multiplicity of (s_i, e_i)). An example of a persistence diagram, generated from Vietoris-Rips filtration of a point cloud representing a noisy circle, is shown in Figure 2.1.3.

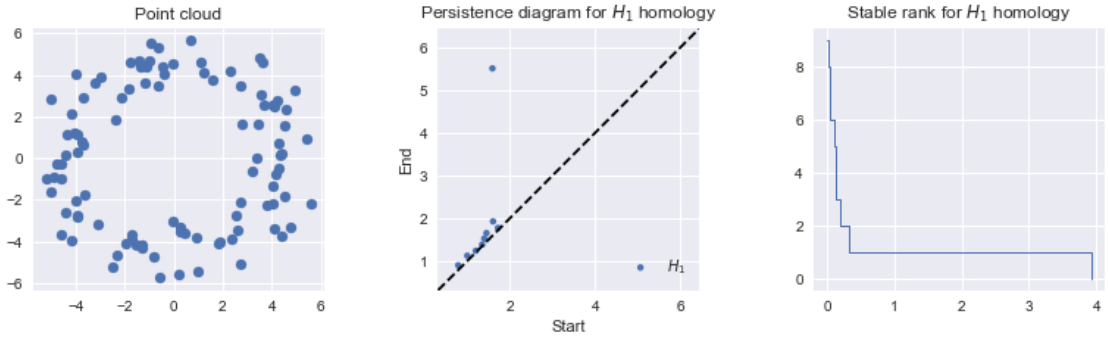


Figure 2.1.3: **Left**: a point cloud representing a noisy circle. **Middle**: the persistence diagram for the H_1 homology resulting from the Vietoris-Rips filtration of the point cloud. **Right**: stable rank with standard contour for H_1 homology resulting from the same filtration.

2.1.7 Pseudometrics on persistence diagrams

We call $\overline{\mathcal{D}}$ the set of extended persistence diagrams, that is composed of elements $d \cup \Delta$ where $d \in \mathcal{D}$ and $\Delta = \{(x, y) \in \mathbb{R}^2 \text{ s.t. } x = y\}$ where each point in Δ has infinite multiplicity.

$\overline{\mathcal{D}}$ can be endowed with the Wasserstein metric defined as:

$$W_p(d_1, d_2) := (\inf_{\gamma} \sum_{x \in d_1} ||x - \gamma(x)||_\infty^p)^{\frac{1}{p}}. \tag{2.6}$$

Where $d_1, d_2 \in \overline{\mathcal{D}}$ and γ is the set of all bijections from d_1 to d_2. Different p correspond to different metrics, for $p = \infty$ it is called the bottleneck metric. The addition of the diagonal can be motivated by observing that points of the diagonal have zero persistence, thus one can say that the information contained in the persistence diagrams is not altered, while such addition makes it possible to define the metric in terms of bijections betweeen sets as those sets now have the same cardinality.

2.2 Stable rank

2.2.1 Contours

A wide class of metrics can also be defined directly on Tame$([0, \infty), \text{Vect})$. We start by defining contours of distance type (for other types of contours see [11]) and will then explain how such contours lead to pseudometrics on Tame$([0, \infty), \text{Vect})$.

We let $f : [0, \infty) \to [0, \infty)$ be a Lebesgue measurable function with strictly positive values, called a density. We further let $C_f(a, \epsilon)$ be the function $C_f : [0, \infty] \times [0, \infty) \to [0, \infty]$ defined by the following integral equation if $a < \infty$:

$$\epsilon = \int_a^{C_f(a,\epsilon)} f(x)dx \tag{2.7}$$

and by $C_f(\infty, \epsilon) := \infty$. Then C_f is called a contour (of distance type) and has the following properties for all a and b in $[0, \infty]$ and ϵ and τ in $[0, \infty)$:

1. If $a \leq b$ and $\epsilon \leq \tau$, then $C_f(a, \epsilon) \leq C_f(b, \tau)$,
2. $C_f(a, 0) = a$,

3. $C_f(C_f(a,\tau),\epsilon) = C_f(a,\tau+\epsilon)$,

4. $C_f(-,\epsilon) : [0,\infty] \to [0,\infty]$ is a monomorphism for every ϵ in $[0,\infty)$,

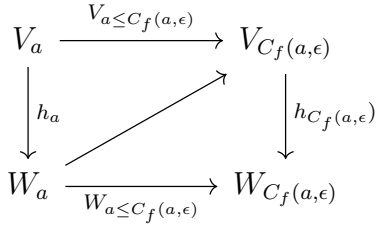
5. $C_f(a,-) : [0,\infty) \to [0,\infty]$ is a monomorphism whose image is $[a,\infty)$ for every a in $[0,\infty)$.

The contour defined by the density $f(x) = 1$ is called the standard contour. We note that the equation above reduces to $\epsilon = \int_a^{C_f(a,\epsilon)} 1dx = C_f(a,\epsilon) - a$ and thus $C_f(a,\epsilon) = a + \epsilon$.

2.2.2 Pseudometrics on tame parametrized vector spaces

For a contour C_f, a pseudometric can now be defined for $V, W \in \text{Tame}([0,\infty), \text{Vect})$, and $\epsilon \in [0,\infty)$ in the following way:

1. A map $h : V \to W$ is called an ϵ-equivalence with respect to C_f if for any $a \in [0,\infty)$ s.t. $C_f(a,\epsilon) < \infty$, there is a linear function $W_a \to V_{C_f(a,\epsilon)}$ making the following diagram commutative:



2. V and W are called ϵ-equivalent with respect to C_f if there is a $X \in \text{Tame}([0,\infty), \text{Vect})$ and maps $g : V \to X \leftarrow W : h$ s.t. g is an ϵ_1-equivalence, h is an ϵ_2-equivalence and $\epsilon_1 + \epsilon_2 \le \epsilon$.

3. Let $S := \{\epsilon \in [0,\infty) \text{ s.t. } V, W \epsilon\text{-equivalent }\}$. Then we define the distance:

$$d_{C_f}(V,W) := \begin{cases} \infty & \text{if } S = \emptyset, \\ \inf(S) & \text{if } S \neq \emptyset. \end{cases} \tag{2.8}$$

Then d_{C_f} is a pseudometric (Proposition 6.3 [11]).

2.2.3 Stable rank

We start by defining the rank of a tame parametrized vector space. For $V \in \text{Tame}([0,\infty), \text{Vect})$ with discretizing sequence $0 < t_0 < ... < t_k$ in $[0,\infty)$ we define $\text{rank}(V) := \dim(H_0(V))$ where:

$$H_0(V) := V_0 \oplus \text{coker}(V_{0<t_0}) \oplus \text{coker}(V_{t_0<t_1}) ... \oplus \text{coker}(V_{t_{k-1}<t_k}). \tag{2.9}$$

The rank is a discrete invariant of V, as its values lie in the natural numbers. We note that it corresponds to the number of bars in a bar decomposition (see 2.1.6). While the rank contains valuable information, the idea of *stable* rank is now to stabilize this information and make it richer: instead of looking simply at the rank of V itself, stable rank takes into account the minimal rank of tame parametrized vector spaces in increasing neighborhoods of V, using the pseudometric d_{C_f} defined in equation 2.8. This process is called hierarchical stabilization of the rank and may also be applied to other discrete invariants. Formally $\widehat{\text{rank}}_{C_f} : \text{Tame}([0,\infty), \text{Vect}) \to \mathcal{M}$ where $\mathcal{M}$ denotes the set of Lebesgue-measurable functions $[0,\infty) \to [0,\infty)$, is defined by:

$$\widehat{\text{rank}}_{C_f}(V)(t) = \min\{\text{rank}(W) \text{ s.t. } W \in \text{Tame}([0,\infty), \text{Vect}) \text{ and } d_{C_f}(V,W) \leq t\}. \tag{2.10}$$

An example of a stable rank function, generated from Vietoris-Rips filtration of a point cloud representing a noisy circle, is shown in Figure 2.1.3. While we will not state all properties of stable ranks here (instead referring directly to [11] in the next sections), we want to highlight two important properties that will be useful:

1. Seen as a map from $\text{Tame}([0,\infty), \text{Vect})$ to a function space endowed with an L_p or interleaving metric, stable rank has a Lipschitz property (hence the name *stable*, Proposition 2.2 [11]), guaranteeing that noise in the input will not be amplified by the mapping.

2. To compute stable rank it suffices to obtain the bar decomposition of V (defined in 2.1.6) for which there are several known algorithms [55]. Once a bar decomposition of V as $\bigoplus_{i=1}^{n} K(s_i, e_i)$ is obtained one can calculate $\widehat{\text{rank}}_{C_f}(V)(t) = \widehat{\text{rank}}_{C_f}(\bigoplus_{i=1}^{n} K(s_i, e_i))(t) = |\{i \text{ s.t. } C(s_i, t) < e_i\}|$ (Proposition 7.2 [11]).

2.2.4 Ample stable rank

Since stable rank is a mapping to a space of functions $f : [0, \infty) \to [0, \infty)$ with rich geometry in which statistical concepts are well-founded, it has the potential to be very useful for statistical analysis and machine learning. In some situations however it is desirable to have an injective map, that is an embedding of tame parametrized vector spaces into a function space. It is clear that for a fixed contour C, stable rank does not have this property. For instance if C is the standard contour, then $\widehat{\text{rank}_C K(s, e)} = \widehat{\text{rank}_C K(s + \delta, e + \delta)}$ for any $\delta > 0$. Thus, two different tame parametrized vector spaces can yield the same stable rank. In fact, when seen through the perspective of bar decomposition and for the standard contour, stable rank only considers the length of bars and not their starting and ending points.

However an embedding into a space $\mathcal{M}_2$ of Lebesgue-measurable functions $[0, \infty) \times [0, \infty) \to [0, \infty)$ can be constructed based on stable rank. We will define $\overline{\text{rank}_{C_f}} : \text{Tame}([0, \infty), \text{Vect}) \to \mathcal{M}_2$, called ample stable rank. Following [11], we start by defining truncation contours. For a fixed contour C_f and $\alpha \in [0, \infty)$ we define another contour, C_f/α as:

$$C_f/\alpha(a, \epsilon) := \begin{cases} C_f(a, \epsilon) & \text{if } C_f(a, \epsilon) < \alpha, \\ \infty & \text{if } C_f(a, \epsilon) \geq \alpha. \end{cases} \tag{2.11}$$

For $V \in \text{Tame}([0, \infty), \text{Vect})$ we can now define $\overline{\text{rank}_{C_f}(V)}$ as the sequence of truncated stable ranks along $\alpha \in [0, \infty)$. That is, for a fixed α, $\overline{\text{rank}_{C_f}(V)(t, \alpha)} = \widehat{\text{rank}_{C_f/\alpha}(V)(t)}$ for all t. When the function space is endowed with an L_p or interleaving distance, this mapping is also shown to have a Lipschitz property (Proposition 2.4 [11]) and constitutes an embedding of isomorphism classes of tame parametrized vector spaces into the function space (Theorem 9.1 [11]). In Figure 2.2.1 an example of ample stable rank is shown. It is generated from the H_1 homology of the point cloud in Figure 2.1.3.

We will sometimes refer to stable rank, $\widehat{\text{rank}_C(V)}$, and ample stable rank, $\overline{\text{rank}_C(V)}$, as feature maps for V, a terminology used in the machine learning literature.

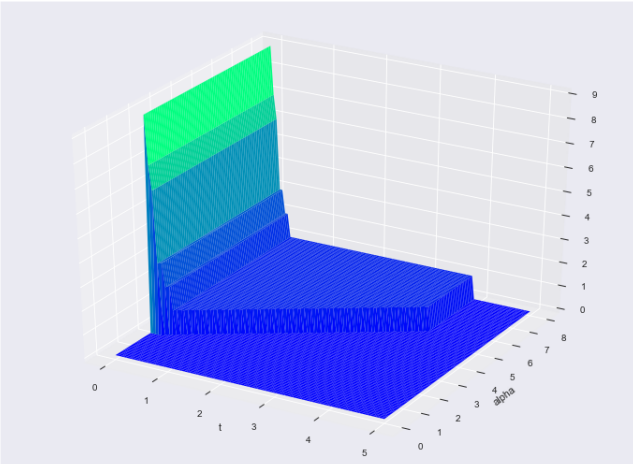


Figure 2.2.1: Example of ample stable rank. Continuation of the example from Figure 2.1.3.

2.3 Statistical approaches for persistent homology

2.3.1 Introduction

In the previous section we have seen how the result of persistent homology can be seen as a persistence diagram endowed with a Wasserstein metric. Alternatively a more algebraic approach can be taken by considering tame parametrized vector spaces endowed with various pseudometrics. From this space, stable rank was developed as a mapping into function spaces. Both approaches naturally lend themselves to data analysis. For instance one can plot and visually investigate persistence diagrams in the plane, or stable ranks as piecewise constant functions, as in Figure 2.1.3. One can also calculate distances between observations in a dataset on which persistent homology has been applied, using the mentioned metrics (in the latter case, distances between stable ranks of two parametrized vector spaces may be conveniently computed in function spaces).

To take the data analysis further it would be useful to take a statistical perspective. One would like to consider probability distributions over the spaces of persistence diagrams or of tame parametrized vector spaces, and be able to infer their properties based on samples. For instance, it would be useful to determine in a precise statistical sense – using the framework of hypothesis testing – whether two samples are drawn from the same underlying distribution. Such distributions over e.g. tame parametrized vector spaces can be seen as being induced by the distribution of the underlying object from which persistent homology is computed. For instance, a distribution over a point

cloud would induce a distribution over the result from the persistent homology of the Vietoris-Rips filtration of such point cloud. A statistical test might thus have the interpretation of determining whether the two distributions have the same topological characteristics.

While applications of such tests are still emerging, a few examples can be found in the literature. Kovacev-Nikolic et al. [29] use a statistical test based on persistent homology to detect conformational changes between closed and open forms of the maltose-binding protein. Kwitt et al. [30] use a statistical test to distinguish between demented and non-demented individuals based on MRI of the corpus callosum region of the brain (a study we will refer to throughout the thesis).

Another interesting application would be to use statistical methods to circumvent some of the computational weaknesses of persistent homology. If for instance we consider a large point cloud on which we want to perform Vietoris-Rips filtration, it would be more efficient to repeatedly uniformly subsample from the point cloud, compute persistent homology and average these results, instead of computing the persistent homology of the original point cloud, since the algorithms to compute persistent homology scale poorly. Of course this is only viable if this averaging is possible and can be shown to approximate the result of persistent homology on the original point cloud in a precise statistical sense. Subsampling is explored in [13].

2.3.2 Overview of approaches

A first possible approach is to consider probability distributions on persistence diagrams. This approach was first explored in [34] where the authors establish Fréchet expectations and variances on the space of persistence diagrams. However Fréchet expectations are not unique which makes the development of statistical methods complicated. Such approaches also face challenges to actually compute the Fréchet means. Turner et al. [48] describe an algorithm for computing the Fréchet mean for a particular class of distributions (combinations of Dirac masses on persistence diagrams). In general the development of such approaches is an active research area.

A second approach is to construct maps from the space of persistence diagrams into spaces where statistical concepts are well-founded, e.g. existence of unique expectation and convergence results that allow for construction of confidence intervals, hypothesis

testing, etc... A persistence landscape [8] constitutes such a map into L^p function spaces, leveraging the theory of probability in Banach spaces to describe e.g. statistical tests.

A third approach is to construct a kernel on the space of persistence diagrams and rely on the theory of kernel embedding of probability distributions to develop statistical methods for persistent homology. The scale-space kernel constructed in this way was introduced in [42] and its statistical properties in the context of kernel embedding of distributions were explored in [30]. Since this kernel is constructed from a feature map into a function space, it has some similarities with the approaches in the previous paragraph (in fact a kernel can be constructed from persistence landscapes, as is done for comparison in [30]). However the theoretical background and statistical methods developed are very different.

Since our approach is inspired by [30], we will devote the next sections to first describe the theory of kernel embedding of distributions after introducing the relevant concepts, and second describe the kernel proposed in the paper.

2.4 Hilbert spaces, RKHS and kernels

2.4.1 Real Hilbert spaces

A real Hilbert space is a real vector space that is equipped with an inner product which is symmetric, bilinear and positive definite. A real Hilbert space is also a complete metric space with respect to the distance induced by its inner product. An example of a real Hilbert space that we will consider in this thesis is the L^2 space of real square integrable functions:

$$f : \mathbb{R} \to \mathbb{R} \text{ s.t. } \int_{-\infty}^{\infty} |f(x)|^2 dx < \infty, \tag{2.12}$$

with the inner product given by:

$$\langle f, g \rangle := \int_{-\infty}^{\infty} f(x)g(x)dx. \tag{2.13}$$

The norm $||f|| = \sqrt{\langle f, f \rangle}$ induces a metric:

$$d(f,g) := ||f-g|| = \sqrt{\langle f,f\rangle + \langle g,g\rangle - 2\langle f,g\rangle} \text{ for } f \neq g. \tag{2.14}$$

2.4.2 Kernels

A symmetric function on an arbitrary non-empty set X, $K : X \times X \to \mathbb{R}$ is called a positive-definite kernel if:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j K(x_i,x_j) \geq 0, \forall x_1,...,x_n \in X \text{ given } n \in \mathbb{N}, c_1,...,c_n \in \mathbb{R}. \tag{2.15}$$

2.4.3 Reproducing Kernel Hilbert Spaces

For a non-empty arbitrary set X, we consider a Hilbert space H of functions $f : X \to \mathbb{R}$. H is said to be a Reproducing Kernel Hilbert Space (RKHS) if there exists an $M > 0$ s.t.:

$$|f(x)| \leq M||f||_H, \forall f \in H. \tag{2.16}$$

An RKHS has the important property that there exists for each $x \in X$ a unique element $K_x \in H$ s.t.

$$f(x) = \langle f, K_x\rangle_H, \forall f \in H. \tag{2.17}$$

In words we can evaluate functions by taking inner products in the RKHS. The reproducing kernel $K : X \times X \to \mathbb{R}$ of H is then defined as $K(x,y) = \langle K_x, K_y\rangle_H$.

The Moore-Aronszajn theorem provides a kind of converse to this since it allows the construction of an RKHS from an arbitrary kernel. Formally: for any kernel K on a set X as defined in 2.4.2, there exists an RKHS of functions $f : X \to \mathbb{R}$ for which K is the reproducing kernel. Starting from a kernel K, we can thus arrive at $K_x = K(x,\cdot)$.

2.5 Kernel embedding of probability distributions

2.5.1 Maximum Mean Discrepancy

We let $\mathcal{X}$ be a topological space on which we define two random variables x and y with Borel probability measures p and q respectively (for which we use the shorthand $x \sim p$ and $y \sim q$). We denote $E_x[f(x)] := E_{x\sim p}[f(x)]$ and $E_y[f(y)] := E_{y\sim q}[f(y)]$ the expectations with respect to p and q respectively. We further let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$. The Maximum Mean Discrepancy (MMD) is defined as:

$$\text{MMD}[\mathcal{F}, p, q] := \sup_{f\in\mathcal{F}}(E_x[f(x)] - E_y[f(y)]). \tag{2.18}$$

It would now be useful to discover function classes $\mathcal{F}$ rich enough for MMD to distinguish probability measures. Formally we want the following to hold:

$$p = q \Leftrightarrow \text{MMD}[\mathcal{F}, p, q] = 0. \tag{2.19}$$

If further an empirical estimate of MMD can be computed from two samples X and Y drawn from p and q respectively (as will be formalized later on), then this estimate may be used as a test statistic in the context of a test for equality of distributions.

2.5.2 MMD and RKHS

While the space $C(\mathcal{X})$ of bounded continuous functions on $\mathcal{X}$ fulfills the condition 2.19 if $(\mathcal{X}, d)$ is a metric space [17], we want to discover function classes that will allow us to practically compute the MMD.

To this aim we now let $\mathcal{H}$ be an RKHS of functions $f : \mathcal{X} \to \mathbb{R}$ with reproducing kernel k, as defined in 2.4.3. Under some conditions $\mathcal{H}$ has the property that a probability distribution p is said to embed in $\mathcal{H}$ through its mean map $\mu_p \in \mathcal{H}$ where μ_p is such that $E_x[f] = \langle f, \mu_p\rangle_{\mathcal{H}}$. This can be thought of as analogue to the reproducing property of the RKHS, allowing us to compute expectations by evaluating inner products in the RKHS. We now state two equivalent conditions for the existence of $\mu_p \in \mathcal{H}$:

1. $k(\cdot,\cdot)$ is measurable and $E_x[\sqrt{k(x,x)}] < \infty$ [24].
2. $k(\cdot,\cdot)$ is measurable and bounded on $\mathcal{X}$ [45].

We now let $\mathcal{F}$ be the space of functions $f \in \mathcal{H}$ s.t. $||f||_{\mathcal{H}} \leq 1$. Assuming the existence of $\mu_p, \mu_q \in \mathcal{H}$ as defined above we are able to express:

$$\begin{aligned} \text{MMD}^2[\mathcal{F}, p, q] &= \left[\sup_{f \in \mathcal{F}} \langle \mu_p - \mu_q, f \rangle_{\mathcal{H}}\right]^2 = ||\mu_p - \mu_q||^2_{\mathcal{H}} \\ &= E_{x,x'}[k(x, x')] + E_{y,y'}[k(y, y')] - 2E_{x,y}[k(x, y)], \end{aligned} \tag{2.20}$$

where x, x' are independent random variables distributed under p, and y, y' independent random variables distributed under q. This offers two perspectives on MMD that will prove to be useful: as the distance between mean maps in $\mathcal{H}$, and as an expression involving expectations of the kernel function.

2.5.3 Universality

We further want a criteria to determine if the condition $p = q$ if.f. $\text{MMD}[\mathcal{F}, p, q] = 0$ holds for $\mathcal{F}$. This is not the case for all RKHS. For instance if we let $\mathcal{X}$ be $\mathbb{R}$ and $\mathcal{H}$ the space of linear maps $f : \mathbb{R} \to \mathbb{R}$ then $\text{MMD}[\mathcal{F}, p, q] = \sup_{a \in \mathbb{R}, a \leq 1} [a(E[x] - E[y])]$, that is it would be sufficient for p and q to have the same mean for $\text{MMD}[\mathcal{F}, p, q]$ to be 0 (while they may still, for example, have different variances).

There is a class of RKHS, called universal, for which the condition holds if $\mathcal{X}$ is a compact metric space. A condition for $\mathcal{H}$ to be universal is for the reproducing kernel $k(\cdot, \cdot)$ to be continuous and for $\mathcal{H}$ to be dense in $C(\mathcal{X})$ with respect to the L_∞ norm.

Another – perhaps more practical – sufficient condition for universality is provided by Christmann and Steinwart [14]. The theorem states that for a compact metric space $\mathcal{X}$, a separable Hilbert space $\mathcal{G}$ and a continuous injective map $\rho : \mathcal{X} \to \mathcal{G}$ the following kernel is universal (that is, the RKHS for which k_U is the reproducing kernel is universal):

$$k_U(x, x') := \sum_{n=0}^{\infty} a_n \langle \rho(x), \rho(x') \rangle^n_{\mathcal{G}}, \text{ for } x, x' \in \mathcal{G}, a_n > 0. \tag{2.21}$$

2.6 Empirical estimates and hypothesis test

2.6.1 Empirical estimates

We now consider a first set $X = \{x_i\}_{i=1}^m$ of observations from a sequence of i.i.d. random variables distributed under p and a second set of observations $Y = \{y_i\}_{i=1}^n$ from a sequence of i.i.d. random variables distributed under q.

A first empirical estimate can be obtained by replacing μ_p by $\frac{1}{m}\sum_{i=1}^m k(x_i, \cdot)$ and μ_q by $\frac{1}{n}\sum_{i=1}^n k(y_i, \cdot)$ in equation 2.20:

$$\begin{aligned} \text{MMD}_b[\mathcal{F}, X, Y] &= ||\frac{1}{m}\sum_{i=1}^m k(x_i, \cdot) - \frac{1}{n}\sum_{i=1}^n k(y_i, \cdot)||_{\mathcal{H}} \\ &= \left[\frac{1}{m^2}\sum_{i,j=1}^m k(x_i, x_j) + \frac{1}{n^2}\sum_{i,j=1}^n k(y_i, y_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(x_i, y_j)\right]^{\frac{1}{2}}. \end{aligned} \tag{2.22}$$

While $\text{MMD}_b[\mathcal{F}, X, Y]$ is the minimum variance estimator [24], it is however biased ($E\left[\text{MMD}_b[\mathcal{F}, X, Y]\right] \neq \text{MMD}[\mathcal{F}, p, q]$) due to the inclusion of terms of type $k(x_i, x_i)$ and $k(y_i, y_i)$. Removing these terms and working with an estimate of MMD^2 we obtain:

$$\text{MMD}_u^2[\mathcal{F}, X, Y] = \frac{1}{m(m-1)}\sum_{i=1}^m\sum_{j\neq i}^m k(x_i, x_j) + \frac{1}{n(n-1)}\sum_{i=1}^n\sum_{j\neq i}^n k(y_i, y_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(x_i, y_j). \tag{2.23}$$

Unbiasedness of $\text{MMD}_u^2[\mathcal{F}, X, Y]$ follows from the fact that it is a U-statistic [44] (on the other hand, $\text{MMD}_b[\mathcal{F}, X, Y]$ is an example of a V-statistic). The choice between $\text{MMD}_b[\mathcal{F}, X, Y]$ and $\text{MMD}_u^2[\mathcal{F}, X, Y]$ is thus an example of the bias-variance tradeoff [20].

Gretton et al. [24] show that both empirical estimates are consistent, that is they converge in probability to the population MMD. Moreover, the convergence rate is shown to be $O((m+n)^{-\frac{1}{2}})$.

In what follows we will only work with $\text{MMD}_b[\mathcal{F}, X, Y]$.

2.6.2 Two-sample hypothesis test

We briefly describe the framework of a two-sample hypothesis test as can be found in [10]. We are in presence of two i.i.d. samples X and Y, drawn from probability distributions p and q respectively. We formulate the two complementary hypotheses:

- $H_0 : p = q$ (null hypothesis),
- $H_a : p \neq q$ (alternative hypothesis).

The goal of the two-sample hypothesis test is now to distinguish between the two hypotheses, based on the samples X, Y. This is done by defining a threshold t (delimiting the acceptance region of the test), such that if $\text{MMD}_b[\mathcal{F}, X, Y] < t$ then H_0 is accepted, otherwise rejected. Two kinds of errors can be made: a Type 1 error is made when $p = q$ is wrongly rejected, while a Type 2 error is made if $p = q$ is wrongly accepted.

While designing the test, a level α is defined, it is an upper bound on the probability of a Type 1 error from which the threshold t is derived. Alternatively a p-value can be computed, which corresponds to the probability, under the null hypothesis, of sampling a test statistic at least as extreme as that which was observed. H_0 can thus be rejected if the p-value does not exceed α.

2.6.3 Distribution of test statistic

In order to use MMD_b as test statistic in two-sample hypothesis test, the distribution under the null hypothesis $p = q$ needs to be understood. Gretton et al. derive an acceptance region for a hypothesis test of level α and where K is s.t. $0 \leq k(x, y) \leq K$ and $m = |X| = |Y|$:

$$\text{MMD}_b[\mathcal{F}, X, Y] < \sqrt{\frac{2K}{m}}(1 + \sqrt{2 \log \alpha^{-1}}). \tag{2.24}$$

This acceptance region may be used for the two-sample hypothesis test. Another alternative, which we will apply in this thesis following [30], is to approximate the distribution of the test statistic under the null hypothesis by bootstrapping on the aggregated data. This procedure is described by Efron and Tibshirani in [18] and for V-statistics in [2]. This distribution is then used to calculate a p-value.

For our samples X and Y such that $|X| = n$, $|Y| = m$, we denote the combined sample of X and Y by Z (thus $|Z| = n + m$). We get the following algorithm:

1. Draw N samples of size $n + m$ with replacement from Z. Call the first n observations X^* and the remaining m observations Y^*.
2. Evaluate $\text{MMD}_b[\mathcal{F}, X^*, Y^*]$ on each sample to obtain $\text{MMD}_b^{(i)}$ for $i = 1, ..., N$.
3. Compute the p-value as $p = \frac{1}{N}\sum_{i=1}^{N} I\{\text{MMD}_b^{(i)} \geq \text{MMD}_b[\mathcal{F}, X, Y]\}$ (where I is the indicator function).

2.7 Scale-space kernel

The scale-space kernel, described in Reininghaus et al. [42], is defined from a feature map $\Phi_\sigma : \mathcal{D} \to L^2(\Omega)$ where $\mathcal{D}$ is the space of persistence diagrams (see 2.1.7) and $\Omega \subset \mathbb{R}^2$ is the closed half-plane above the diagonal. Inspired by scale-space theory [28], the feature map is the solution to a heat diffusion differential equation, defined by setting as initial conditions a sum of Dirac deltas on the points in the persistence diagram and a Dirichlet boundary condition on the diagonal. Once the feature map is obtained by solving the differential equation, a kernel is defined as the inner product in $L^2(\Omega)$ and the kernel can be expressed in closed-form for $F, G \in \mathcal{D}$:

$$k_\sigma(F, G) := \langle \Phi_\sigma(F), \Phi_\sigma(G) \rangle_{L^2(\Omega)} = \frac{1}{8\pi\sigma} \sum_{p \in F, q \in G} e^{\frac{-||p-q||^2}{8\sigma}} - e^{\frac{-||p-\bar{q}||^2}{8\sigma}}. \tag{2.25}$$

The parameter σ can be seen as controlling the scale of the kernel. The points $p, q = (s, e)$ are points of the persistence diagrams and $\bar{q} = (e, s)$, the point mirrored in the diagonal.

The kernel is shown to be stable with respect to the Wasserstein distance (see 2.1.7) for $p = 1$. By considering probability distributions on a subset of $\mathcal{D}$ on which some conditions are imposed (all points in the diagrams are start/end bounded, as well as the total multiplicity of the diagrams) it is further shown that the RKHS $\mathcal{H}$ for which k_σ is the reproducing kernel has the property that probability distributions can be said to embed in $\mathcal{H}$ through their mean map (see 2.5.2), justifying the application of statistical methods from the theory of kernel embedding of distributions to the space of persistence diagrams. A kernel derived from k_σ and defined as $k_\sigma^U(F, G) :=$

$\exp(k_\sigma(F, G))$ is further shown to be universal (see 2.5.3).

2.8 Machine learning

2.8.1 Kernel methods

We will briefly introduce kernel methods for machine learning and make the connection to the background on kernels in section 2.4. One way to understand the usefulness of kernels is to start from the formulation of support-vector machines (SVM), a well-known kernel method. If we take the setting of a binary classification, we have a training set $\{(x_i, y_i)\}_{i=1}^N$ where $x_i \in \mathbb{R}^{d_1}$ is a vector of predictors and $y_i \in \{0, 1\}$. Instead of the original predictors we can choose to work with $\phi(x_i)$, an arbitrary feature map $\phi : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$. SVM now proposes to find a decision boundary that is a hyperplane in the feature space (for the formulation of the optimization problem see [20]). Solving the optimization problem, it appears that SVM predicts the class y^* for an out of sample observation x^* as:

$$y^* = \text{sgn}(\sum_{i=1}^{N} w_i y_i \langle \phi(x_i), \phi(x^*) \rangle), \tag{2.26}$$

where w_i are coefficients obtained by solving the optimization problem. Thus the prediction (and the solution of the optimization problem) only requires inner products in the feature space. This opens up to work with feature maps to an arbitrary Hilbert space $\mathcal{H}$ and not just Euclidean, i.e. $\phi : \mathbb{R}^{d_1} \to \mathcal{H}$, something that will be exploited in this thesis. Further, as any kernel as defined in 2.4.2 realizes an inner product in some (possibly unknown) Hilbert space, one is not required to construct $\phi(x_i)$ explicitly, but can directly construct a kernel $k(x_i, x_j)$ (satisfying the conditions in 2.4.2) which can be seen as a similarity function between two observations (this is the so-called kernel trick).

2.8.2 Artificial neural networks

A (feedforward) artificial neural network [23] can be seen as a particular class of statistical models for a conditional probability $P(Y_i = y_i | \Theta = \theta, X_i = x_i)$ where $\{(X_i, Y_i)\}$ are independent pairs of random variables (X_i are predictors, Y_i the label)

and Θ parameters. For instance for a binary classification problem where $y_i \in \{-1, 1\}$ one has:

$$P(Y_i = y_i|\Theta = \theta, X_i = x_i) = \text{sigm}(y_i f_\theta(x_i)), \tag{2.27}$$

where $f_\theta : \mathbb{R}^d \to \mathbb{R}$ (in the case of binary classification problem for a d-dimensional input vector) is s.t. $f_\theta = f^{(n)} \circ ... \circ f^{(1)}$, i.e. a composition of functions of type $f^{(i)}(h^{(i-1)}) = g^{(i)}(W^{(i)}h^{(i-1)} + b^{(i)})$. We say that $f^{(i)}$ represent transition functions between layers $h^{(i-1)}$ and $h^{(i)}$ ($h^{(0)}$ being the input layer of the neural network and $h^{(n)}$ the output layer), $W^{(i)}, b^{(i)} \subset \theta$ are parameters and $g^{(i)}$ is called an activation function (e.g. ReLu or sigmoid) and is applied component-wise.

For a training set $\{(x_i, y_i)\}_{i=1}^N$ (that is, outcomes of $\{(X_i, Y_i)\}_{i=1}^N$) one typically computes $\hat{\theta}$ by maximum likelihood of $P(Y_i = y_i|\Theta = \theta, X_i = x_i)$ on the training set (using the backpropagation algorithm). Predictions for out-of-sample observations are computed based on this estimate of θ.

After this brief introduction we will now focus on aspects of neural networks that will be of particular importance for the models developed in this thesis.

2.8.3 Sets as inputs to neural networks

In the previous section we considered the input of a neural network to be a vector in $\mathbb{R}^d$, which is the general assumption. We will now consider instead neural networks where the input is a set. We can formulate this by first defining a set $\mathcal{X}$. Then the power set $2^\mathcal{X}$ is the input domain of our neural network. A point cloud is an example of such an input, where $\mathcal{X} = \mathbb{R}^d$ and an input $X \in 2^\mathcal{X}$ is thus a set of variable cardinality (though in practice finite) of points, i.e. vectors in $\mathbb{R}^d$. Point clouds are for instance produced by 3D-scanners.

Formally we can say that our function f_θ defined in the previous section for the example of binary classification is now a function $f_\theta : 2^\mathcal{X} \to \mathbb{R}$ with the property that if $M \in \mathbb{N}$, for any permutation π on $\{1, ..., M\}$ the following holds: $f_\theta(x_1, ..., x_M) = f_\theta(x_{\pi(1)}, ..., x_{\pi(M)})$. We then call f_θ permutation-invariant.

While particular forms of permutation-invariant neural networks had been developed, as well as more ad-hoc methods (e.g. adding permuted versions of the data to the

training set, a form of data augmentation), Zaheer et al. [53] proposed a general form for a permutation-invariant network. They show that f_θ is permutation-invariant if.f. it can be decomposed in the form:

$$f_\theta(X) = \rho(\sum_{x \in X} \phi(x)), \tag{2.28}$$

where ρ and ϕ are suitable transformations. The sum may also be replaced by other permutation-invariant operations (e.g. max, median...). In fact Zaheer et al. only prove this theorem for the case when $\mathcal{X}$ is countable (which is not the case for point clouds for instance), but it is conjectured to hold in general.

Hence a general type of architecture for a neural network is defined where from the input layer every member of the set $x \in X$ is transformed by $\phi(x)$ (possibly through several layers) and this output, for all members of the set, then goes through a permutation-invariant operation such as a sum. There may be several such transformations and their concatenation is then fed into the subsequent part of the network, ρ, which can be defined by several layers.

2.9 Persistent homology and machine learning

2.9.1 Introduction

We now move to investigating the connection between persistent homology and machine learning. In this thesis we will concern ourselves with supervised learning, thus our aim can be seen as developing methods where the information about the topology and geometry of observations in a dataset – as may be extracted with persistent homology – is exploited to attain a lower loss in a particular machine learning problem, for instance classifying observations with high accuracy. While we notice the commonalities between machine learning and statistical methods such as those described earlier (e.g. the analogy between binary classification and statistical tests to distinguish samples from two distributions) we treat them separately and develop different methods (however relying on the same feature map or kernel) in this thesis. While our focus is supervised learning, we note that there are several other interesting intersections between persistent homology and machine learning, which constitute active research areas. One of them is generative models, where persistent

homology may be used to learn representations that are topologically accurate, and less sensitive to adversarial attacks [35]. Another example is topological regularization, where parameters in e.g. a neural network can be forced (or encouraged through priors) to adopt a particular topology, for instance to form clusters [7]. A more specific application of this idea is to encourage connectivity in image segmentation through topological priors [15].

Supervised learning using purely input obtained from the persistent homology method has been shown to be competitive for a few machine learning problems. Within graph learning, when the problem is to classify observations solely based on their graph structure (without any data attached to the edges or vertices) methods based on persistent homology can achieve high accuracy, as is explored in the experiment in section 5.2.2, following [27]. However the most promising route is likely to develop methods that successfully exploit the information extracted by persistent homology as one kind of feature, alongside with other features obtained from the observations in a dataset. It is believed that topological features are often uncorrelated to features that can be extracted through non-topological methods and thus can enhance models in several areas. An illustration of such approach is in Dindin et al. [16] where a neural network is constructed to detect arrhythmia in patients from ECG time series. Persistent homology on the sublevel filtration of the time series (through the use of Betti curves [49]) is used as one input to the neural network, alongside other features derived from the time series. An ablation study is performed, showing that inclusion of topological features increased the accuracy of the classification by several percentage points. It has also been observed [7] that many adversarial attacks in computer vision are of topological nature, which may indicate that convolutional neural networks – which are highly effective at e.g. image recognition – by their strategy of utilizing local connectivity aggregated at higher levels through the convolution operation, may still miss global, topological, characteristics of images.

2.9.2 Overview of approaches

Persistence diagrams, which are typically seen as the topological summaries resulting from persistent homology, do not easily lend themselves to the construction of machine learning methods, due to similar reasons as those discussed in section 2.3.2 and to the fact that a metric structure is generally not enough for the development of

machine learning methods, in which the structure given by Euclidean spaces, or more generally Hilbert spaces, is desirable. In fact, most of the research done to utilize persistent homology in a machine learning context can be seen through this lens: as ways to develop stable mappings with high discriminative power from the space of persistence diagrams to either Euclidean spaces (for neural networks, sometimes called vectorization) or more generally Hilbert spaces (for kernel methods). We will review some of the previous work in those two categories.

A first approach for vectorization is provided by Adams et al. [1] under the name persistence images. For a persistence diagram B rotated clock-wise by $\frac{\pi}{4}$, the function $\rho_B : \mathbb{R}^2 \to \mathbb{R}$ is defined as:

$$\rho_B(z) = \sum_{p \in B} \phi_p(z) \cdot w(p), \tag{2.29}$$

where ϕ_p is taken to be the Gaussian centered at p, and w is a function that can be used to weigh areas of the persistence diagram differently. Thus intuitively $\rho_B(z)$ gives a measure of the density of points of the persistence diagram in the neighborhood of z. This is further vectorized by fixing a grid and integrating $\rho_B(z)$ over each square of the grid.

A first approach for a kernel constructed from a mapping into a Hilbert space is the scale-space kernel [42] described in section 2.7.

Persistence Landscapes [8] are defined for a persistence diagram B as piecewise linear functions $\lambda_B : \mathbb{Z}_+ \times \mathbb{R} \to \mathbb{R}$. The building block is a "tent" function $\lambda_p(t)$ defined for each point $p = (s, e)$ of the persistence diagram as:

$$\lambda_p(t) := \begin{cases} t - s & \text{if } t \in [s, \frac{s+e}{2}], \\ e - t & \text{if } t \in [\frac{s+e}{2}, e], \\ 0 & \text{otherwise.} \end{cases} \tag{2.30}$$

From this $\lambda_B(k, t) = \text{kmax}_{p \in B} \lambda_p(t)$ is defined, where kmax is the k:th largest value in the set. Persistence landscapes have been applied both for vectorization by discretizing $\lambda_B(k, t)$, and as a kernel method: since λ_B can be seen as a mapping into L^2, a kernel can be constructed from the inner product in this space.

2.9.3 Learnable parameters

While a mapping from persistence diagrams into function spaces may be injective, as we saw was the case for the the ample stable rank (2.2.4) and for the feature map from which the scale-space kernel (2.7) is derived, the discretization induced by a vectorization (to $\mathbb{R}^n$) will intuitively mean that some data is preserved while some is lost. A natural question is thus to ask whether this choice can be made explicit. Some methods indeed give some control over this choice by parametrizing the vectorization, as was the case for the persistence images defined in equation 2.29, where the weighing function allowed to incorporate priors over which areas of the diagram are thought to be the most important for a particular task. However, since designing such priors is hard, efforts have recently concentrated on producing vectorizations that are not only parametrizable, but where those parameters can be learned during the training of a neural network. Typically this is done by ensuring that the loss function defined by a neural network is differentiable with respect to the parameters, so that those can be learned during backpropagation together with the other parameters of the network. Such methods can thus claim to provide a task-specific (for a particular learning problem, e.g. for a fixed dataset, loss function and network design) optimal vectorization, that is optimal among the parametrizable family of functions considered.

The first such approach by Hofer et al. [27] has similarities with both the scale-space kernel and persistence images. The building block is a Gaussian-like function $S_{\mu,\sigma,\nu}(s,e)$ applied on a point (s,e) of a persistence diagram B rotated clock-wise by $\frac{\pi}{4}$, where ν is fixed but the parameters $\mu = (\mu_0, \mu_1)$ and $\sigma = (\sigma_0, \sigma_1)$ are designed to be learned during the training phase. It is defined as:

$$S_{\mu,\sigma,\nu}(s,e) := \begin{cases} \exp\{-\sigma_0^2(s-\mu_0)^2 - \sigma_1^2(e-\mu_1)^2\} & \text{if } e \in [\nu, \infty), \\ \exp\{-\sigma_0^2(s-\mu_0)^2 - \sigma_1^2(\ln\frac{e}{\nu} + \nu - \mu_1)^2\} & \text{if } e \in (0, \nu), \\ 0 & \text{if } e = 0. \end{cases} \tag{2.31}$$

From this building block, a vectorization (into $\mathbb{R}$) is obtained for B by summing over all points in the persistence diagram: $\sum_{(s,e)\in B} S_{\mu,\sigma,\nu}(s,e)$. This function being differentiable with respect to μ and σ, it can serve as the first layer of a neural network. For a richer representation, n pairs of parameters (μ, σ) can be initialized, leading to

a learnable representation as a fixed-size vector $\mathbb{R}^n$. A follow-up paper [26] considers other building blocks, which may have better numerical stability properties than the Gaussian-like one.

Learnable representations are further generalized in PersLay [9], which inspired by permutation-invariant networks [53] described in section 2.8.3, defines a generic form for an input layer of a neural network as:

$$\text{PERSLAY}(B) := \text{op}(\{w(p) \cdot \phi(p)\}_{p \in B}), \tag{2.32}$$

where op is a permutation-invariant operation (such as sum, max, etc...), w is a weighing function (such as the one defined for persistence images) and ϕ is a parametrizable function which maps each point of the diagram into a vector. For instance for the construction above $\phi = S_{\mu,\sigma,\nu}$, op is the sum operation, and $w(p) = 1$.

2.9.4 Stability

Most mappings are constructed to be stable, that is they have a Lipschitz property, which was seen for example for stable rank (2.2.3). This quest can also be seen in the construction by Hofer et al. described in equation 2.31, where the transformation of points within a fixed distance ν of the diagonal is necessary to fulfill Lipschitz-continuity with respect to the Wasserstein metric on the space of persistence diagrams with $p = 1$. Such transformations, which lead to points of low persistence being discounted compared to other points, corresponds to a prior of considering such points as noise, or at least of less importance for learning problems. This is however not true in general, as commented on in [11].

Chapter 3

Kernel-based learning and statistical analysis

3.1 Introduction

In this section we will define kernels based on stable rank, and describe how they can be used in the context of kernel methods in machine learning. We then proceed to describe the reproducing kernel Hilbert space (RKHS) that can be derived from the kernels and show that they have the property that probability measures on the underlying space of tame parametrized vector spaces can be said to embed in this RKHS. We will examine some of the properties of the kernels such as universality. Finally we will describe a two-sample hypothesis test based on the kernels.

The methodology borrows from Reininghaus et al. [42] and Kwitt et al. [30] in that the goal is to develop first a kernel, second a statistical method for persistent homology using kernel embedding of distributions. The kernels are however constructed in different ways since they are based on different feature maps. This construction needs to be described and mathematically motivated, which is the goal of this section. Once established, the computational and discriminative properties can be examined, with the hope of providing an alternative to the method based on the kernel described in the two cited papers, and in general to other machine learning and statistical methods based on persistent homology.

3.2 Restrictions on parametrized vector spaces

We consider $\text{Tame}([0,\infty), \text{Vect})$, the collection of tame, $[0,\infty)$-parametrized vector spaces, as defined in 2.1.5. We further consider a fixed contour C of distance type (as defined in 2.2.1). We now define $\text{Tame}_{T,N}([0,\infty), \text{Vect})$ as the collection of elements $V \in \text{Tame}([0,\infty), \text{Vect})$ where V is s.t:

1. There is a $T \in [0,\infty)$ such that $V_a = 0$ for all $a > T$.

2. $\text{rank}(V) \leq N$ ($N \in \mathbb{N}$).

3.3 Interpretation of the restrictions

While imposing such bounds is necessary for the theoretical development, they do not in practice constitute limitations. To illustrate this, we consider applying Vietoris-Rips on a point cloud, i.e. a finite metric space (X,d) with $|X| = M$ points. Such point cloud has $\binom{M}{2}$ pairwise distances, which could be ordered to obtain a sequence $d_1, ..., d_{\binom{M}{2}}$ such that $d_i \leq d_{i+1}$. $d_{\binom{M}{2}}$ is thus the maximum distance between any two points, called the diameter of the point cloud. From the definition of Vietoris-Rips (see 2.1.4) $\text{VR}_t(X,d) = \text{VR}_v(X,d)$ for $d_{\binom{M}{2}} \leq t \leq v$, and taking the corresponding tame parametrized vector space V, $V_{t\leq v}$ is an isomorphism. If necessary, one can thus without loosing any information "truncate" V by setting $V_v = 0$ and $V_{t\leq v}$ be the map from V_t to 0, for $t < d_{\binom{M}{2}} \leq v$.

We also see that for $d_i \leq t \leq v < d_{i+1}$ we have $\text{VR}_t(X,d) = \text{VR}_v(X,d)$. Thus the minimal discretizing sequence (see 2.1.5) τ of V must be such that $|\tau| \leq \binom{M}{2}$. By the definition of the rank of V in terms of dimension of direct sums of cokernels (2.2.3) we must have $\text{rank}(V) \leq \dim(V_{d_1}) + ... + \dim(V_{d_{\binom{M}{2}}})$. Recalling the definition of homology from 2.1.3 we can further find a bound on $\dim(V_t)$ based on the number of n-simplices, e.g. for H_0 homology the bound is the maximum number of 0-simplices, i.e. M. Hence the rank in that case is bounded by $M \cdot \binom{M}{2}$.

One can further consider M and $d_{\binom{M}{2}}$ derived from e.g. storage capacity of computers and not from a particular dataset.

3.4 Constructing the kernels

For a chosen contour C we will construct two kernels: K^1_C and K^2_C. We do this by first showing that $\widehat{\text{rank}_C(V)}$ and $\overline{\text{rank}_C(V)}$ (defined in 2.2.3 and 2.2.4) are square-integrable for $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$. We then define the kernels as inner-products in L^2 (L^2 was defined as an example of a Hilbert space in 2.4). The references to e.g. definitions and propositions in the following proofs are found in [11].

Proposition 2. *$\widehat{\text{rank}_C(V)} \subset L^2(\mathbb{R})$ for $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$.*

Proof. By Definition 2.1 $\widehat{\text{rank}_C(V)}$ is a Lebesgue-measurable function $[0,\infty) \to [0,\infty)$. We need to show that $\widehat{\text{rank}_C(V)}$ is square-integrable. By Definition 7.1, $\widehat{\text{rank}_C(V)}$ is a non-increasing piecewise constant function. It is bounded because $\widehat{\text{rank}_C(V)}(0) = \text{rank}(V) \leq N$ by Corollary 7.9 and condition 2 in 3.2 (we note however that this holds even without the bound on the rank since the rank is finite for any tame parametrized vector space). Since $\widehat{\text{rank}_C(V)}$ is a piecewise constant function, requiring it to be square-integrable is equivalent to requiring it to have bounded support (for otherwise $\lim_{t\to\infty} \widehat{\text{rank}_C(V)}(t) > 0$ and it can not be square-integrable).

We consider the decomposition of V as $\oplus_{i=1}^n K(s_i, e_i)$. From condition 1 in 3.2 we know that $e_i \leq T$, $\forall i$. Following Proposition 8.2 we have that $\text{life}_C K(s_i, e_i) = C(s_i, -)^{-1}(e_i)$. From the properties of regular contours stated in 2.2.1 we must have that $\text{life}_C K(s_i, e_i) = C(s_i, -)^{-1}(e_i) \leq C(0, -)^{-1}(T)$. For otherwise assume $L_i = C(s_i, -)^{-1}(e_i) > C(0, -)^{-1}(T) = L_T$. Then $e_i = C(s_i, L_i) > C(0, L_T) = T$ (by Definition 1 $(s_i, L_i) \geq C(0, L_T)$ but since the contour is regular $C(-, \epsilon)$ and $C(a, -)$ are mononorphisms hence $(s_i, L_i) > C(0, L_T)$) which contradicts condition 1 in 3.2. We define $\ell_{\max} = C(0, -)^{-1}(T)$.

From what we established and Corollary 8.3 we have that $\widehat{\text{rank}_C(V)}(t) = 0$ for $t > \ell_{\max}$. Thus $\widehat{\text{rank}_C(V)}$ is square-integrable. □

Proposition 3. *$\overline{\text{rank}_C(V)} \subset L^2(\mathbb{R}^2)$ for $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$ and $\alpha \in [0, T]$.*

Proof. In 2.2.4 we saw that $\overline{\text{rank}_C(V)}$ is such that for a fixed α, $\overline{\text{rank}_{C_f}(V)(t, \alpha)} = \widehat{\text{rank}_{C_f/\alpha}(V)}(t)$ for all t.

It holds for $\alpha \leq T$ that $\text{life}_{C/\alpha}(s_i, e_i) \leq \text{life}_C(s_i, e_i)$ (Proposition 8.2). For a fixed α we thus have that $\overline{\text{rank}_C(V)}$ has finite support $[0, \ell_{\max}]$ and is therefore square-integrable.

Since we consider $\alpha \in [0, T]$, it thus holds that $\overline{\text{rank}_C(V)(t, \alpha)}$ is bounded, has finite support $[0, \ell_{\max}] \times [0, T]$ and is therefore square-integrable, i.e. an element of $L^2(\mathbb{R}^2)$. □

Proposition 4. *$K_C^1, K_C^2 : Tame_{T,N}([0,\infty), Vect) \times Tame_{T,N}([0,\infty), Vect) \to \mathbb{R}$ defined by:*

- $K_C^1(V, W) := \int_0^\infty \widehat{rank_C(V)(t)}\widehat{rank_C(W)(t)}dt$,
- $K_C^2(V, W) := \int_0^T (\int_0^\infty \overline{rank_C(V)(t, \alpha)}\ \overline{rank_C(W)(t, \alpha)}dt)d\alpha$,

are kernels as defined in 2.4.2.

Proof. The functions are by definition inner products in the Hilbert space L^2 of the feature maps of two elements in $\text{Tame}_{T,N}([0,\infty), \text{Vect})$. Any inner product is symmetric and positive-definite. □

3.5 Stability of the kernels

3.5.1 First kernel

Using the definition in 2.4.1, the norm derived from the inner product in L^2 induces a pseudometric on $f, g \in L^2$, hence on $\widehat{\text{rank}_C(V)}, \widehat{\text{rank}_C(W)}$ for $V, W \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$, called the L^2-distance:

$$d_{L^2}(\widehat{\text{rank}_C(V)}, \widehat{\text{rank}_C(W)}) = (\int_0^\infty |\widehat{\text{rank}_C(V)(t)} - \widehat{\text{rank}_C(W)(t)}|^2 dt)^{\frac{1}{2}}. \tag{3.1}$$

Proposition 2.2 [11] establishes the following stability result:

$$d_{L^2}(\widehat{\text{rank}_C(V)}, \widehat{\text{rank}_C(W)}) \leq c\, d_C(V, W)^{\frac{1}{2}}, \tag{3.2}$$

where d_C is the pseudometric on $\text{Tame}([0,\infty), \text{Vect})$ with respect to the contour C as defined in 2.2.3 and $c = \max\{\widehat{\text{rank}_C(V)(0)}, \widehat{\text{rank}_C(W)(0)}\}$.

Since the kernel is defined as the inner product in L^2, stability of K_C^1 is equivalent to the stability of the feature map thus shown.

3.5.2 Second kernel

In Proposition 2.4 [11] the normalized L_p metric is defined, for $V, W \in \text{Tame}([0,\infty), \text{Vect})$:

$$\widehat{L_p}(\overline{\text{rank}_C(V)}, \overline{\text{rank}_C(W)}) := \lim_{a\to\infty} \frac{1}{a} \int_0^a \Big(\int_0^\infty |\overline{\text{rank}_C(V)}(\alpha,t) - \overline{\text{rank}_C(W)}(\alpha,t)|^p dt \Big)^{\frac{1}{p}} d\alpha, \tag{3.3}$$

and shows the following stability result:

$$c\, d_C(V,W)^{\frac{1}{p}} \geq \widehat{L_p}(\overline{\text{rank}_C(V)}, \overline{\text{rank}_C(W)}), \tag{3.4}$$

where $c = \max\{\widehat{\text{rank}_C(V)(0)}, \widehat{\text{rank}_C(W)(0)}\}$.

Adapting this proposition to the case $p = 2$ and considering our restrictions $\alpha \in [0, T]$ we obtain for $V, W \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$:

$$\widehat{L_2}(\overline{\text{rank}_C(V)}, \overline{\text{rank}_C(W)}) = \frac{1}{T} \int_0^T \Big(\int_0^\infty |\overline{\text{rank}_C(V)}(\alpha,t) - \overline{\text{rank}_C(W)}(\alpha,t)|^2 dt \Big)^{\frac{1}{2}} d\alpha. \tag{3.5}$$

3.5.3 Comment on the stability of kernels

We note that our kernels are additive, that is for $V, W, Z \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$ they have the property $k(V \oplus W, Z) = k(V, Z) + k(W, Z)$. This follows from the property $\widehat{\text{rank}_C}(V \oplus W) = \widehat{\text{rank}_C}(V) + \widehat{\text{rank}_C}(W)$ (Theorem 6.2 in [11]) and from the bilinearity of the inner product.

Reininghaus et al. [42] prove the following theorem for non-trivial kernels (a trivial kernel is such that $k(V, W) = 0$ for all V, W) defined on persistence diagrams endowed with the Wasserstein metric $d_{W,p}$:

Proposition 5. *A non-trivial additive kernel k on persistence diagrams is not Lipschitz-stable with respect to $d_{W,p}$ for any $1 < p \leq \infty$.*

Because of the proximity between the pseudometric defined from the standard contour (2.2.1) and the interleaving metric [43], and the isometry theorem [31] showing that the interleaving pseudometric is equal to bottleneck distance (that is the Wasserstein distance for $p = \infty$) it is not surprising that this result holds in our case too. Indeed for $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$, we take $V^n = V \oplus ... \oplus V$ the direct sum of n copies of V. We thus have:

$$d_{L^2}(\widehat{\text{rank}_C(V^n)}, 0) = \sqrt{k(V^n, V^n) + k(0,0) - 2k(V^n, 0)} = n\sqrt{k(V,V)}. \tag{3.6}$$

On the other hand $d_C(V^n, 0) = d_C(V, 0)$, thus there is no constant c such that we can bound $d_{L^2}(V^n, 0)$ by $c \cdot d_C(V^n, 0)$ for every V and n.

It is thus not possible to have a Lipschitz-continuous kernel. However one could argue that in practice the kind of stability stated in section 3.5 suffices: indeed the constant is $\max\{\widehat{\text{rank}_C(V)(0)}, \widehat{\text{rank}_C(W)(0)}\}$, that is the maximum of the rank of the two tame parametrized vector spaces. For a particular experiment such a bound is easy to derive. On the other hand, not restricting oneself to $d_{W,1}$ opens up for a richer class of kernels with potentially superior discrimative power.

3.6 Kernel methods for machine learning

To summarize what has been derived so far, we now have kernels K_C^1 and K_C^2 based on the stable rank for a chosen contour, for which stability results hold. We note that a rich collection of kernels thus can be obtained by choosing different contours, suggesting the possibility to find a contour that would prove superior to others in terms of discriminative power for a particular learning problem.

The kernels can be used in kernel methods for machine learning such as SVM as described in 2.8.1. Moreover, any kernel satisfies the requirements of a covariance function, thus our kernel can also be used in the context of e.g. Gaussian Process regression or classification.

3.7 Computing the kernel

Because stable ranks of tame parametrized vector spaces are piecewise constant functions, K_C^1 is particularly efficient to compute in closed form. We can describe $\widehat{\text{rank}_C(V)}$ as a sequence discretizing its domain: $P^V = \{t_i\}$ s.t. $0 = t_0 \leq t_1 \leq ... \leq t_{n-1} \leq t_n$ and an associated vector of values M^V where $M_i^V = \widehat{\text{rank}_C(V)}(t), t_{i-1} \leq t < t_i$. If we now have two elements $V, W \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$ together with their stable ranks we call $P^{V,W} = P^V \cup P^W$ the union of the discretizing sequences and M^V, M^W the vectors of stable rank values of V, W corresponding to this discretization. We can thus compute the kernel K_C^1 as: $\sum_{i=1}^{N}(t_i - t_{i-1})M_i^V M_i^W$ for $t_i \in P^{V,W}$ where $N = |P^{V,W}|$. This indicates that computing the kernel K_C^1 based on its stable rank scales linearly in the size of the bar decomposition. Obtaining the stable rank for a particular contour from e.g. a bar decomposition is itself an efficient operation as described in Section 9 in [11].

3.8 RKHS and probability embedding

Having established a kernel K_C^1, by the Moore-Aronszajn theorem stated in 2.4.3 we now also have an RKHS of functions $f : \text{Tame}_{T,N}([0,\infty), \text{Vect}) \to \mathbb{R}$ for which K_C^1 is the reproducing kernel. We call this space $\mathcal{H}$. Explicitly, $\mathcal{H}$ is generated by the linear span of $\{K_C^1(V, \cdot)$ s.t. $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})\}$ together with its completion. The same argument holds for K_C^2.

We now consider a random variable x with Borel probability measure p on $\text{Tame}_{T,N}([0,\infty), \text{Vect})$. Referring to section 2.5.2 we would now like to establish whether p is embedded into $\mathcal{H}$ through its mean map $\mu_p \in \mathcal{H}$ where μ_p is such that $E_x[f] = \langle f, \mu_p \rangle_{\mathcal{H}}$.

Proposition 6. *K_C^1, K_C^2 are bounded.*

Proof. In 3.4 we established that the support of $\widehat{\text{rank}_C(V)}$ is $[0, \ell_{\max}]$. From the fact that $\widehat{\text{rank}_C(V)}$ is non-increasing we know that it attains its maximum for $\widehat{\text{rank}_C(V)}(0)$. By Definition 7.3 and Corollary 7.8 [11] we know that $\widehat{\text{rank}_C(V)}(0) = \text{rank}(V) \leq N$ for all $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$ because of the restrictions defined in 3.2. Since the kernel is just the integral of the product of two bounded functions on a bounded interval K_c^1

will itself be bounded.

Similarly, $\overline{\text{rank}_C(V)}$ has the same upper bound as $\widehat{\text{rank}_C(V)}$. The domain being also bounded, K_C^2 is also bounded. □

Proposition 7. *K_C^1, K_C^2 are measurable.*

Proof. From [45] Proposition 2, the condition is that all $f : \text{Tame}_{T,N}([0,\infty), \text{Vect}) \to \mathbb{R}$, defined by first fixing $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$, then taking $f = K_C^1(V, \cdot)$, are measurable. Such a function is the composition of:

- The feature map, i.e. $\widehat{\text{rank}_C}$. Its continuity is implied by the stability results discussed in 3.5.
- The inner product in the feature space L^2, when holding one argument constant. Any inner product is continuous when holding one argument constant.

As a composition of two continuous functions, f is continuous, which implies that it is measurable. The same argument applies to K_C^2. □

Together, the boundedness and measurability of the kernel imply the existence of $\mu_p \in \mathcal{H}$ by the condition stated in 2.5.2.

3.9 Universality

We now turn to discussing universality of the kernel, a concept introduced in 2.5.3. We first note that for a fixed contour C, K_C^1 can not be universal. The argument is similar to that of non-injectivity of stable rank in 2.2.4: let C be the standard contour, and $V, W \in \text{Tame}([0,\infty), \text{Vect})$ s.t. V is a unique bar $K(s,e)$ and W is another unique bar $K(s+\delta, e+\delta)$, $\delta > 0$. We now have probability measures p and q where p puts a single point mass (Dirac measure) on V and q a point mass on W. Then $\mu_p = \widehat{\text{rank}_C(V)} = \widehat{\text{rank}_C(W)} = \mu_q$. Thus $\text{MMD}[\mathcal{H}, p, q] = 0$ despite $p \neq q$.

The second kernel, K_C^2, which is built from an injective feature map, seems to be a better candidate. We recall from 2.2.4 that this feature map is $\overline{\text{rank}_C} : \text{Tame}([0,\infty), \text{Vect}) \to \mathcal{M}_2$ where $\mathcal{M}_2$ denotes the set of Lebesgue-measurable functions $[0,\infty) \times [0,\infty) \to [0,\infty)$. When considering isomorphism classes of $\text{Tame}([0,\infty), \text{Vect})$, this map is injective (Theorem 9.1 [11]).

Since $\text{Tame}_{T,N}([0,\infty),\text{Vect}) \subset \text{Tame}([0,\infty),\text{Vect})$ the map $\text{Tame}_{T,N}([0,\infty),\text{Vect}) \to \mathcal{M}_2$ is also injective. In 3.4, to obtain square-integrable functions in $\mathcal{M}_2$ we restricted ourselves to functions $[0,\infty) \times [0,T] \to [0,\infty) \subset \mathcal{M}_2$ which we call $\mathcal{M}_2^*$. We claim that the map $\text{Tame}_{T,N}([0,\infty),\text{Vect}) \to \mathcal{M}_2^*$ is also injective. For $\mathcal{M}_2^*$ we consider a sequence of stable ranks along $\alpha \in [0,T]$ instead of $\alpha \in [0,\infty)$. We know that for a bar decomposition of $V \in \text{Tame}_{T,N}([0,\infty),\text{Vect})$ as $\bigoplus_{i=1}^n K(s_i,e_i)$ we have that $e_i \leq T$ for all i. Thus from Proposition 8.2 [11] if $\alpha \geq T$ it holds that $\text{life}_{C/\alpha}(s_i,e_i) = \text{life}_C(s_i,e_i)$. From the characterization of stable rank as $\widehat{\text{rank}_{C_f}}\left(\bigoplus_{i=1}^n K(s_i,e_i)\right)(t) = |\{i | t < \text{life}_{C_f} K(s_i,e_i)\}|$ this implies that $\widehat{\text{rank}_{C/\alpha_1}(V)} = \widehat{\text{rank}_{C/\alpha_2}(V)}$, if $\alpha_1,\alpha_2 \geq T$. In other words, $\overline{\text{rank}_C(V)(t,\alpha)}$ is constant for $\alpha \geq T$. Hence the restriction of $\overline{\text{rank}_C}$ to $\alpha \in [0,T]$, i.e. the map $\text{Tame}_{T,N}([0,\infty),\text{Vect}) \to \mathcal{M}_2^*$ must also be injective.

It is however unclear if K_C^2 is universal. Following [30] and using the theorem stated in 2.5.3 we can however show that a kernel derived from it is universal:

Proposition 8. *The kernel* $K_C^U(V,W) := \exp\left(K_C^2(V,W)\right)$ *is universal.*

Proof. First we note that K_C^U is of the form $\sum_{n=0}^{\infty} a_n \langle \rho(x), \rho(x') \rangle_{\mathcal{G}}^n$, where $a_n = \frac{1}{n!} > 0$, that is it corresponds to the Taylor expansion of $\exp\left(K_C^2(V,W)\right)$ and ρ is the feature map $\overline{\text{rank}_C(V)}$ into $L^2(\mathbb{R}^2) = \mathcal{G}$, a separable Hilbert space. Furthermore the stability result from 3.5 implies that the feature map is continuous.

Together with the compactness of $\text{Tame}_{T,N}([0,\infty),\text{Vect})$, to which we devote the next section, we have shown the assumptions necessary to apply the theorem stated in 2.5.3, showing universality of K_C^U. □

3.10 Compactness

We recall from 2.2.1 that to define a contour of distance type we start with a Lebesgue measurable function $f : [0,\infty) \to [0,\infty)$ with strictly positive values, called a density. Then $C_f : [0,\infty] \times [0,\infty) \to [0,\infty]$ is defined by the following integral equation:

$$\epsilon = \int_a^{C_f(a,\epsilon)} f(x)dx. \tag{3.7}$$

We now consider a class of densities $\mathcal{D}$ defined instead on a closed subset of $[0,\infty)$. For instance based on the restrictions in 3.2 one can choose $[0,T]$, since the values of f on

(T, ∞) will have no impact on the stable rank obtained from the contour C_f. We also require densities in $\mathcal{D}$ to be continuous.

The goal is now to show compactness with respect to the pseudometric induced by an arbitrary contour C_f of distance type, for $f \in \mathcal{D}$. We start by showing that all such pseudometrics are topologically equivalent. Two pseudometrics d_1, d_2 on $\mathcal{X}$ are topologically equivalent if they generate the same topology on $\mathcal{X}$. Since compactness is a topological property then $(\mathcal{X}, d_1)$ is compact if.f. $(\mathcal{X}, d_2)$ is compact. Instead of showing topological equivalence we will show the stronger condition of strong equivalence of pseudometrics. Two pseudometrics d_1, d_2 on $\mathcal{X}$ are strongly equivalent if there exists positive constants α, β such that for every $x, y \in \mathcal{X}$:

$$\alpha d_1(x, y) \leq d_2(x, y) \leq \beta d_1(x, y). \tag{3.8}$$

Proposition 9. *Consider two densities $f, g \in \mathcal{D}$. Then the pseudometrics d_{C_f}, d_{C_g} induced by contours of distance type for the densities f and g are strongly equivalent.*

Proof. We call X the closed subset on which f and g are defined. By the extreme value theorem, f and g attain their minimum and maximum on X. Hence there exist non-zero constants α and β such that $\alpha g(x) \geq f(x) \geq \beta g(x), \forall x$.

These inequalities imply that $C_{\alpha g}(a, \epsilon) \leq C_f(a, \epsilon), \forall a, \forall \epsilon$.

By definition $C_g(a, \epsilon)$ is s.t. $\int_a^{C_g(a,\epsilon)} g(x)dx = \epsilon$. We have:

$$\int_a^{C_g(a,\epsilon)} g(x)dx = \epsilon \Leftrightarrow \alpha \int_a^{C_g(a,\epsilon)} g(x)dx = \alpha\epsilon \Leftrightarrow \int_a^{C_g(a,\epsilon)} \alpha g(x)dx = \alpha\epsilon, \tag{3.9}$$

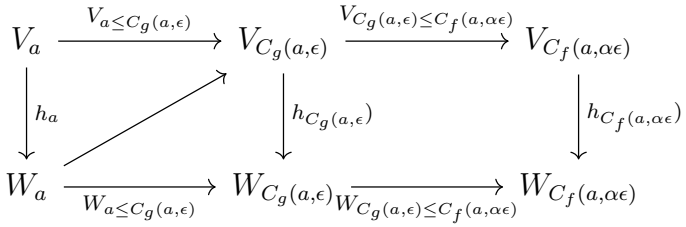
which is the equation defining $C_{\alpha g}(a, \alpha\epsilon)$. Hence $C_g(a, \epsilon) = C_{\alpha g}(a, \alpha\epsilon)$.

Putting our two previous results together we get:

$$C_g(a, \epsilon) \leq C_f(a, \alpha\epsilon), \forall a, \forall \epsilon. \tag{3.10}$$

This implies that every $h : V \to W$ which is an ϵ-equivalence with respect to C_g is also an $\alpha\epsilon$-equivalence with respect to C_f. This is shown in the following diagram where the linear function $W_a \to V_{C_g(a,\epsilon)}$ making the diagram commute assures that h is an ϵ-equivalence with respect to C_g and the composition of this function with $V_{C_g(a,\epsilon) \leq C_f(a,\alpha\epsilon)}$

assures that h is also an $\alpha\epsilon$-equivalence with respect to C_f.



Assume $d_{C_g}(V,W) = \epsilon$. This means there exists X s.t. $h : V \to X \leftarrow W : \ell$ where h is an ϵ_1-equivalence, ℓ is an ϵ_2-equivalence with respect to C_g and $\epsilon_1 + \epsilon_2 \leq \epsilon$.

By our result above we have that h is an $\alpha\epsilon_1$-equivalence and ℓ an $\alpha\epsilon_2$-equivalence with respect to C_f. Since $\alpha\epsilon_1 + \alpha\epsilon_2 \leq \alpha\epsilon$ we have that V, W are $\alpha\epsilon$-equivalent with respect to C_f. In conclusion it thus holds that:

$$d_{C_f}(V,W) \leq \alpha\epsilon = \alpha d_{C_g}(V,W). \tag{3.11}$$

Using the same arguments we can show that $d_{C_f}(V,W) \geq \beta d_{C_g}(V,W)$ for some β. Hence d_{C_f} and d_{C_g} are strongly equivalent. □

All pseudometrics induced by contours of distance type for densities in $\mathcal{D}$ are thus topologically equivalent. The constant density $f(x) = 1$ is in this class of densities. The contour defined by this density is called the standard contour (2.2.1). Hence all pseudometrics d_{C_f} for f in $\mathcal{D}$ are topologically equivalent to the pseudometric induced by the standard contour d_C.

This pseudometric has a special relationship with another pseudometric considered in the literature on parametrized vector spaces: the interleaving distance, introduced in [12]. It is shown in [43] that d_C, the pseudometric induced by the standard contour, is strongly equivalent to the interleaving distance.

We have already seen and used the fact that tame parametrized vector spaces are bar decomposable, thus can be represented as persistence diagrams. In 2.1.7 we defined the Wasserstein distances on the space of persistence diagrams. A natural question is thus: if we take two tame parametrized vector spaces V, W, and their persistence diagrams B_V, B_W, what is the relationship between between $d_I(V,W)$ (the interleaving distance defined on parametrized vector spaces) on the one hand, and $d_{W_p}(B_V, B_W)$ on the other hand (the Wasserstein distance on persistence diagrams)?

The isometry theorem [31] shows that in fact $d_I(V, W) = d_{W_\infty}(B_V, B_W)$, that is the interleaving distance on parametrized vector spaces is equal to the bottleneck distance (the Wasserstein distance for $p = \infty$) on their persistence diagrams.

Thus, for topological questions such as compactness, we are justified in moving to considering the representation of our tame parametrized vector spaces as persistence diagrams, endowed with the bottleneck distance, in which the literature is more developed.

We can start by reinterpreting the restrictions T, N on $\text{Tame}_{T,N}([0, \infty), \text{Vect})$ now considered as persistence diagrams, which we can call $\mathcal{S}_{T,N}$. First, we have already seen that for $S \in \mathcal{S}_{T,N}$ the first condition in 3.2 implies that for all $x = (s, e) \in S$ it holds that $e \leq T$. This implies that both the end and the start ($s < e$) of points in the persistence diagrams are bounded. Second, the bound on $\text{rank}(V)$ for $V \in \text{Tame}_{T,N}([0, \infty), \text{Vect})$ is equivalent to a bound on the total multiplicity $|S|$ of $S \in \mathcal{S}_{T,N}$, as the rank coincides with the smallest number of elements generating V, that is the total multiplicity of the persistence diagram.

A complete characterization of relatively compact spaces with respect to the bottleneck distance is provided in [40]. Closer to our case, Kwitt et al. [30] prove that $\mathcal{S}_{T,N}$ is compact with respect to the Wasserstein distance with $p = 1$. It thus only remains to show that $\mathcal{S}_{T,N}$ is also compact with respect to the Wasserstein distance with $p = \infty$, that is the bottleneck distance. We can show that the two distances, d_{W_1} and d_{W_∞} are strongly equivalent. This would imply that compact spaces with respect to d_{W_1} are also compact with respect to d_{W_∞}.

For an arbitrary bijection from persistence diagram d_1 to d_2 $\gamma \in \Gamma$, the set of all such bijections, it holds that $\sum_{x \in d_1} ||x - \gamma(x)||_\infty \geq \sup_{x \in d_1}\{||x - \gamma(x)||_\infty\}$. It thus also holds that $\sum_{x \in d_1} ||x - \gamma(x)||_\infty \geq \inf_{\gamma \in \Gamma} \sup_{x \in d_1}\{||x - \gamma(x)||_\infty\}$ for all $\gamma \in \Gamma$. Thus also $d_{W_1}(d_1, d_2) = \inf_{\gamma \in \Gamma} \sum_{x \in d_1} ||x - \gamma(x)||_\infty \geq \inf_{\gamma \in \Gamma} \sup_{x \in d_1}\{||x - \gamma(x)||_\infty\} = d_{W_\infty}(d_1, d_2)$. In the other direction it holds for an arbitrary $\gamma \in \Gamma$ that $2N \cdot \sup_{x \in d_1}\{||x - \gamma(x)||_\infty\} \geq \sum_{x \in d_1} ||x - \gamma(x)||_\infty$. Thus $2N \cdot d_{W_\infty}(d_1, d_2) = 2N \cdot \inf_{\gamma \in \Gamma} \sup_{x \in d_1}\{||x - \gamma(x)||_\infty\} \geq \inf_{\gamma \in \Gamma} \sum_{x \in d_1} ||x - \gamma(x)||_\infty = d_{W_1}(d_1, d_2)$. This concludes the proof of compactness of $\text{Tame}_{T,N}([0, \infty), \text{Vect})$ with respect to distances induced by densities in the class defined in the beginning of this section.

3.11 Two-sample hypothesis test

In 3.8 we defined RKHS of functions $f : \mathrm{Tame}_{T,N}([0,\infty), \mathrm{Vect}) \to \mathbb{R}$ from our kernels K_C^1, K_C^2 and showed that probability distributions on $\mathrm{Tame}_{T,N}([0,\infty), \mathrm{Vect})$ embed in those spaces through their mean maps. The same can be done for K_C^U, moreover the RKHS obtained from this kernel is universal (3.9).

We can use the space of functions thus constructed to define MMD as in 2.5.2, from which empirical estimates can be obtained (2.6.1). While other statistical tests can be conceived based on this kernel embedding of probability distributions, we will in this thesis use the two-sample hypothesis test as illustration. Adapting the methodology of 2.6.2 to our setting, we thus obtain a statistical test capable of determining whether two samples, obtained by the method of persistent homology, can be said to come from the same probability distribution over tame parametrized vector spaces.

Chapter 4

Neural network input layer

4.1 Introduction

In section 3.6 we introduced a first method for machine learning based on stable rank, namely kernel methods such as SVM. While such methods are often highly effective, in our context they suffer from two main weaknesses.

The first weakness is the computational complexity. For a training set $\{(x_i, y_i)\}_{i=1}^N$, kernel methods rely on computing the kernels $K(x_i, x_j)$ for all $i, j = 1, ..., N$, that is $O(N^2)$ kernel computations. Hence, even when the kernel itself can be efficiently computed as described in 3.7, such methods become infeasible for very large training sets.

Second, and somewhat more specific to our application, we explained in 2.2.1 how different contours (defined by densities) give rise to different pseudometrics on tame parametrized vector spaces, which in turn give rise to different stable ranks. It is very likely that some contours lead to kernels that are better than others at e.g. discriminating between classes for a particular dataset. It would thus be very valuable to discover which contours are optimal for a particular learning problem, however this is not obvious in the context of kernel methods. We will develop the idea of learnable contour in the next sections and explain why it warrants a shift from kernel methods to neural networks.

4.2 Parametrized families of contours

In 2.2.1 we defined contours derived from a density f, which is a strictly positive function $f : [0, \infty) \to [0, \infty)$. A first step towards learnable contours is to restrict ourselves to contours defined by a parametrized family of densities. While many such choices are possible, we will in this thesis work with a univariate Gaussian, parametrized by μ, restricted to the domain $[0, \infty)$ (as an extension, the standard deviation σ may also constitute a parameter):

$$f_\mu(x) := \frac{1}{\sigma\sqrt{2\pi}} \exp\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\}. \tag{4.1}$$

We note that while f_μ restricted to the domain $[0, \infty)$ is no longer a probability distribution, it still fulfills the conditions for a density as defined here.

4.3 Stable rank for parametrized families

Mathematically, stable ranks are defined by a process of hierarchical stabilization of the rank, a discrete invariant of tame parametrized vector spaces (see 2.2.3). To actually calculate the stable rank the bar decomposition of $V \in \text{Tame}_{T,N}([0, \infty), \text{Vect})$ as $\bigoplus_{i=1}^n K(s_i, e_i)$ is used. Practically, algorithms for computing persistent homology (as implemented by e.g. Ripser [47]) provide the bar decomposition, called barcode, as a set $X = \{(s_i, e_i)\}_{i=1}^n$ which we can consider as the input to our machine learning algorithm.

As we move to a such an algorithmic view of stable rank, the following characterization (Proposition 8.1 [11]) will be useful:

$$\widehat{\text{rank}_{C_f}}(V)(t) = \widehat{\text{rank}_{C_f}}\Big(\bigoplus_{i=1}^n K(s_i, e_i)\Big)(t) = |\{i | t < \text{life}_{C_f} K(s_i, e_i)\}|. \tag{4.2}$$

By our first restriction in 3.2, V is such that there is a $T \in [0, \infty)$ s.t. $V_a = 0$ for all $a > T$. This implies that in the bar decomposition $e_i < \infty$, for all i. Hence by Proposition 8.2 [11] we have:

$$\text{life}_{C_f} K(s_i, e_i) = C_f(s_i, -)^{-1}(e_i). \tag{4.3}$$

For C_{f_μ}, the contour based on our Gaussian density, we have:

$$\text{life}_{C_f} K(s_i, e_i) = C_{f_\mu}(s_i, -)^{-1}(e_i) = \int_{s_i}^{e_i} f_\mu(x)dx. \tag{4.4}$$

Looking at the definition of our kernels in 3.4 we thus see that since μ is a parameter of stable ranks it also appears as parameter of the kernel. One could thus consider methods for (hyper)parameter optimization of kernels, the simplest of which is a grid search: for a selection of M parameters $\{\mu_i\}_{i=1}^M$ we simply compute the kernels and fit the training set (e.g. solve the SVM optimization problem on the training set) M times. The parameter with the lowest error rate on a validation set could then be kept.

More sophisticated ways to learn optimal parameters are not obvious in SVM. For Gaussian processes, model selection might be a possible route, that is optimizing parameters based on the marginal likelihood of the dataset [41], however this was not explored in this thesis. Instead we will move to the setting of a neural network.

4.4 Towards neural network input layers

We would like to move towards defining an input layer with learnable parameter μ for a neural network as defined in 2.8.3, that is for a set $X = \{(s_i, e_i)\}_{i=1}^n$ resulting from the bar decomposition of $V \in \text{Tame}_{T,N}([0, \infty), \text{Vect})$.

We note that if we consider the stable rank of V and fix t we get by equation 4.2 and 4.3 an element:

$$\widehat{\text{rank}_{C_{f_\mu}}}(V)(t) = |\{i | t < C_{f_\mu}(s_i, -)^{-1}(e_i)\}| \in \mathbb{N}. \tag{4.5}$$

This corresponds to the format defined for a permutation-invariant network in equation 2.28, where $X = \{(s_i, e_i)\}_{i=1}^n$, $\phi(x) = C_{f_\mu}(s_i, -)^{-1}(e_i)$, and counting the number of $\phi(x)$ over a certain threshold t is a permutation-invariant operation. Discretizing at several values $t_1, ..., t_d$ and repeating this procedure thus leads to a vector in $\mathbb{N}^d$ that can serve as input for further layers of the network.

The problem however is the discrete nature of $\widehat{\text{rank}_{C_{f_\mu}}}(V)(t)$ that makes it hard to formulate an optimization problem where the parameters of the density f_μ could be optimized.

One possibility could be to use a smooth variant of stable rank, as suggested in [21]:

$$\widetilde{\text{rank}}_{C_{f_\mu}}\Big(\bigoplus_{i=1}^{n} K(s_i, e_i)\Big)(t) := \sum_{(s_i,e_i)\in X} \frac{1}{1+\exp\{k(t - C_{f_\mu}(s_i,-)^{-1}(e_i))\}}. \tag{4.6}$$

The smooth stable rank, for a fixed t, is a sum of logistic functions. Intuitively if $C_{f_\mu}(s_i,-)^{-1}(e_i) >> t$ the value is close to 1. If on the other hand $C_{f_\mu}(s_i,-)^{-1}(e_i) << t$ then the value is close to 0. In this way the function approximates the stable rank. The parameter k controls the smoothness. The smooth stable rank is also permutation-invariant and converges to stable rank as $k \to \infty$. Contrary to stable rank, it is however differentiable with respect to parameters in the density as long as $C_{f_\mu}(s_i,-)^{-1}(e_i))$ is differentiable. Therefore the smooth stable rank could be a good candidate for an input layer to a neural network with learnable contours. In this thesis we will however choose another approach, perhaps leading to a simpler network architecture.

4.5 Discretizing the y-axis

We start with a slight change of perspective: instead of discretizing $\widehat{\text{rank}}_{C_{f_\mu}}(V)$ along the x-axis we propose to discretize along the y-axis. Stable rank is not injective, i.e. the preimage $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y) = \{t \in [0,\infty) \text{ s.t. } \widehat{\text{rank}}_{C_{f_\mu}}(V)(t) = y\}$, for $y \in \mathbb{N}$ is not unique, however we can make the convention:

$$\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y) := \sup\{t \in [0,\infty) \text{ s.t. } \widehat{\text{rank}}_{C_{f_\mu}}(V)(t) = y\}. \tag{4.7}$$

Because $\widehat{\text{rank}}_{C_{f_\mu}}(V)$ is a non-increasing piecewise-constant function, $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}$ is particularly well-behaved for $V \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$, it is in fact also non-increasing.

We can give the following interpretation to this function. We consider the set $\{\text{life}_{C_{f_\mu}} K(s_i,e_i)\}_{i=1}^n$. We recall that if C was the standard contour then this set would be the persistence of all bars in the decomposition, that is $\{e_i - s_i\}_{i=1}^n$. By analogy it may be helpful to think of $\{\text{life}_{C_{f_\mu}} K(s_i,e_i)\}_{i=1}^n$ as a kind of generalized measure of the persistence of V, where contours allow us to discriminate among topological features appearing at different time scales by applying different weights (represented by the

density).

We let $y_{\max} = \widehat{\text{rank}}_{C_{f_\mu}}(V)(0)$, which corresponds to the maximum of $\widehat{\text{rank}}_{C_{f_\mu}}(V)$. We then have that $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(1) = \max\{\text{life}_{C_{f_\mu}} K(s_i, e_i)\}_{i=1}^n$, while $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y_{\max}) = \min\{\text{life}_{C_{f_\mu}} K(s_i, e_i)\}_{i=1}^n$ and in general:

$$\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y) = \text{y:th biggest value } \{\text{life}_{C_{f_\mu}} K(s_i, e_i)\}_{i=1}^n. \tag{4.8}$$

In Figure 4.5.1 the process is illustrated. The top left plot shows a barcode $\{(s_i, e_i)\}_{i=1}^n$ on which a Gaussian density f is superposed. For the contour based on this density, $\{\text{life}_{C_f} K(s_i, e_i)\}_{i=1}^n$ is calculated and shown in the top right plot. The indexing for those two plots is the same, thus for a particular bar in the left plot, one can see the corresponding "life length" of this bar in the right plot, at the same vertical level. The sorted set (from biggest to smallest) $\{\text{life}_{C_f} K(s_i, e_i)\}_{i=1}^n$ is shown in the bottom left plot. In this example we have $y_{\max} = n = 13$. We wish to extract $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(7)$, that is the median value of $\{\text{life}_{C_f} K(s_i, e_i)\}_{i=1}^n$. This corresponds to the seventh value in the sorted set as shown by the arrow. In the bottom right plot the stable rank under the contour C_f is shown, where the dotted line represents $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(7)$, which corresponds to the same value as in the previous plot.

4.6 Properties of the discretization

The observations in the previous section suggest a simple algorithm for calculating $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}$. Starting from a bar decomposition $X = \{(s_i, e_i)\}_{i=1}^n$ we calculate $\{\text{life}_{C_{f_\mu}} K(s_i, e_i)\}_{i=1}^n$ with equation 4.4. We then transform this set into an n-dimensional vector P by ordering the values from biggest to smallest. Formally $P = [C_{f_\mu}(s_{\pi(1)}, -)^{-1}(e_{\pi(1)}), C_{f_\mu}(s_{\pi(2)}, -)^{-1}(e_{\pi(2)}), ..., C_{f_\mu}(s_{\pi(n)}, -)^{-1}(e_{\pi(n)})]$, where π is a permutation on the indices $1, 2, ..., n$ s.t. $P_i \geq P_{i+1}, \forall i$. Then $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y) = P_y$.

We further note that $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}$ has the format defined for a permutation-invariant network from equation 2.28 where $\phi(x) = C_{f_\mu}(s_i, -)^{-1}(e_i)$ for $x = (s_i, e_i) \in X$, and the permutation-invariant operation is now the k:th biggest value of a set. Finally, because of its proximity to stable rank, we can hope that $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}$ can be discriminative in a supervised learning setting.

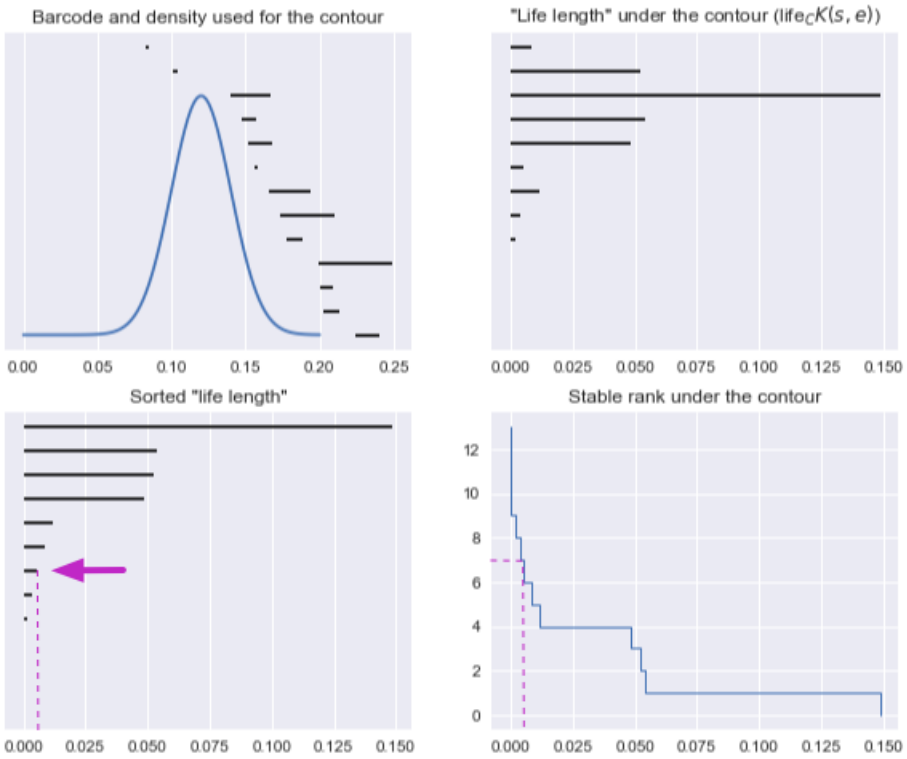


Figure 4.5.1: **Top left**: a barcode with a density f superposed. **Top right**: the corresponding "life length" of the bars under the contour C_f. **Bottom left**: the same set of "life lengths" but sorted. **Bottom right**: the stable rank under the contour C_f.

The benefit of using $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}$ however becomes visible when remembering that we are looking for a function that is differentiable with respect to our parameters. In section 4.4 we saw that the discrete nature of stable rank made this difficult. However for our newly defined function this is now easy.

First we note that we can get the derivative of $C_{f_\mu}(s_i, -)^{-1}(e_i)$ with respect to μ from equation 4.4. This is true for any differentiable density f. In our case we can easily get the expression in closed form by interchanging the order of derivation and integration:

$$\begin{aligned} \frac{d}{d\mu} C_{f_\mu}(s_i, -)^{-1}(e_i) &= \varphi(e_i) - \varphi(s_i), \\ \text{where } \varphi(x) &= -\frac{\sqrt{\sigma}}{\sqrt{2\pi}} e^{\frac{-(x-\theta)^2}{2\sigma^2}}. \end{aligned} \tag{4.9}$$

We have that $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y)$ is a composition of $C_{f_\mu}(s_i, -)^{-1}(e_i)$ and the y:th biggest value operation. If the y:th biggest value is unique the derivative is:

$$\frac{d}{d\mu}\text{y:th biggest value}\{C_{f_\mu}(s_i,-)^{-1}(e_i)\}_{i=1}^n = \frac{d}{d\mu}C_{f_\mu}(s_*,-)^{-1}(e_*), \\ \text{where } * = \arg\text{y:th biggest value } \{C_{f_\mu}(s_i,-)^{-1}(e_i)\}_{i=1}^n. \tag{4.10}$$

In the case when the y:th biggest value operation is not unique then $\widehat{\text{rank}}_{C_{f_\mu}}(V)^{-1}(y)$ is in fact not differentiable. However, even at these points of nondifferentiability there exists a subgradient set. Thus optimization algorithms such as stochastic subgradient descent can be applied to this problem.

4.7 Constructing neural network architectures

We have thus constructed a subdifferentiable input layer based on stable rank. Moreover the construction has the potential to be efficient since it only requires keeping the index of the y:th biggest value during the forward pass of the neural network. Then in the backpropagation step, computation of the derivative $\frac{d}{d\mu}C_{f_\mu}(s_i,-)^{-1}(e_i)$ is only required on one element of the set $X = \{(s_i,e_i)\}_{i=1}^n$.

Our input layer needs one more ingredient: we remember that from persistent homology and bar decomposition of observations (e.g. finite metric spaces) of a dataset, the cardinality of the set can vary, i.e. $X = \{(s_i,e_i)\}_{i=1}^n$ may have different n for every observation. We can accommodate for this by instead working with percentiles of the set $\{\text{life}_{C_{f_\mu}}K(s_i,e_i)\}_{i=1}^n$. For instance a discretization designed to extract a feature vector in $\mathbb{R}^2$ could correspond to extracting the 25%:th and 75%:th percentile of the set.

Now a complete neural network can be constructed by adding several elements, e.g. discretizing $[0,100]$ are regular intervals and computing y:th percentile for all such values. One can also work with several parametrizations, e.g. $\mu_1,\mu_2,\ldots$. Finally the input layer may be combined with any neural network architecture for the subsequent layers (e.g. the number of hidden layers, units per layers, regularization techniques can be chosen freely). This is illustrated in Figure 4.7.1.

We also note that neural networks may be constructed where the input layer corresponds to one of many channels. For instance a neural network may combine the input layer developed here with another input layer corresponding to some non-

topological features extracted from the same input. The network could thus learn a function where those different types of features interact in the subsequent hidden layers.

4.8 Implementation

The neural network input layer is implemented in PyTorch [38] as a custom autograd class. Its forward function takes a multiset $X = \{(s_i, e_i)\}_{i=1}^{n}$ of variable length n as input and returns a vector in $\mathbb{R}^{d \cdot p}$ where d is the number of parameters and p the number of features (percentiles) that the layer is configured to extract for each parameter μ. Its backward function returns the gradient with respect to the parameters. Instances of this autograd class can now be combined with the building blocks of PyTorch, e.g. linear layers, activation functions, loss functions and optimizers to easily build complex networks.

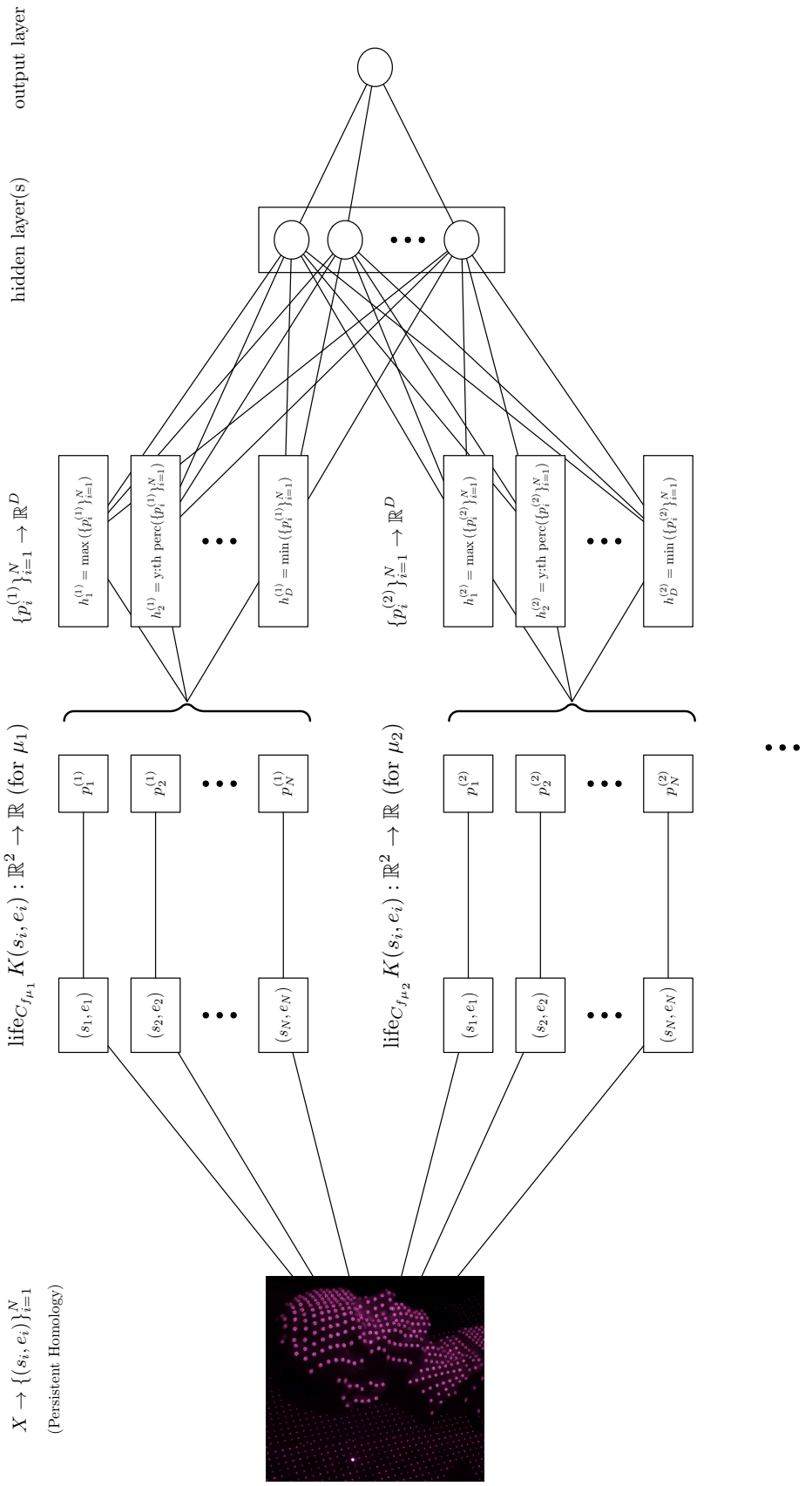


Figure 4.7.1: Example of neural network architecture based on stable rank input layer.

Chapter 5

Experiments

5.1 Kernel-based statistical test

5.1.1 Geometric shapes

Data

The first experiment is related to topological features of point clouds sampled from geometric shapes with noise; it will illustrate the procedure of the kernel-based two-sample hypothesis test. We start with three geometric shapes: the circle, the pine and the square. From these geometric shapes we will generate point clouds (finite subsets of $\mathbb{R}^2$) in the following way: we uniformly sample 100 points from a shape and add Gaussian noise. Examples of such point clouds can be seen in Figure 5.1.1. We generate samples of n such point clouds from each shape, collected in $\mathcal{T}_c, \mathcal{T}_p, \mathcal{T}_s$ for circle, pine and square respectively. For instance $\mathcal{T}_c = \{X_i\}_{i=1}^n$, where X_i is a point cloud sampled from the circle. We will concern ourselves with the H_1 homology from a Vietoris-Rips filtration. To this aim, persistent homology is applied to each point cloud, after which stable rank (with standard contour) is computed. Stable ranks are plotted underneath the example point clouds in Figure 5.1.1. Stable rank for H_1 homology and the standard contour can be interpreted as showing how long different H_1 homology classes persist. For the circle and the square we can interpret the stable rank plot as capturing that there is one class (representing the hole) that persists for a long time (until around $x = 15$), whereas for the pine there are only classes with very short persistence ($x < 0.2$), formed locally from the noise in the sample.

In the experiments we vary the standard deviation of the Gaussian noise added to the points sampled from the shapes, in order to illustrate the impact of noise. The effect of the added noise on a point cloud is illustrated in Figure 5.1.2. The number of point clouds in each sample is also varied.

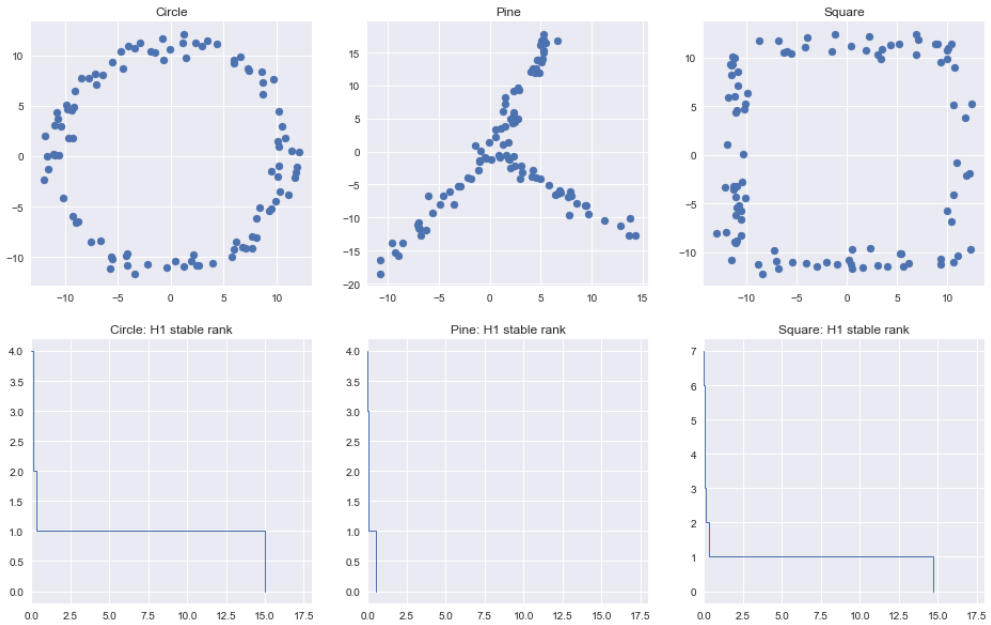


Figure 5.1.1: Examples of point clouds sampled from a circle, a pine and a square, together with H_1 stable rank.

Figure 5.1.2: Point clouds generated from a circle, sampled with various levels of noise.

Method

The probability distribution of the noisy geometric shape induces a probability distribution over the tame parametrized vector spaces $\mathrm{Tame}_{T,N}([0,\infty), \mathrm{Vect})$ that are the results of the H_1 persistent homology procedure over the noisy geometric shape.

Thus for a first test we would like to consider whether the probability distribution c over

Noise (σ) / Sample size (n)	20	40	60	80	100
2	0	0	0	0	0
4	0	0	0	0	0
6	0.0003	0.0043	0	0	0
8	0.0210	0.0058	0	0	0
10	0.1017	0.0720	0.0484	0.0087	0.0179

Table 5.1.1: P-values obtained from hypothesis test of *circle vs. pine* for various levels of noise (rows) and sample size (columns).

Noise (σ) / Sample size (n)	20	40	60	80	100
2	0.0670	0.0076	0.0005	0	0
4	0.1417	0.0353	0.0078	0.0007	0
6	0.2209	0.0576	0.0351	0.0079	0.0042
8	0.2902	0.1212	0.0140	0.0401	0.0040
10	0.2359	0.1427	0.0875	0.0868	0.0186

Table 5.1.2: P-values obtained from hypothesis test of *circle vs. square* for various levels of noise (rows) and sample size (columns).

$\text{Tame}_{T,N}([0,\infty), \text{Vect})$ induced by the noisy circle equals the probability distribution p induced by the pine. Formally our test can be expressed as:

- H_0: $c = p$ (null hypothesis),
- H_a: $c \neq p$ (alternative hypothesis).

Equipped with our (first) kernel, defined in 3.4, and the samples $\mathcal{V}_c$ and $\mathcal{V}_p$ where $\mathcal{V}_c = \{V_i\}_{i=1}^n$, $\mathcal{V}_p = \{W_i\}_{i=1}^n$, $V_i, W_i \in \text{Tame}_{T,N}([0,\infty), \text{Vect})$, i.e. the parametrized vector spaces computed from the sample of point clouds $\mathcal{T}_c, \mathcal{T}_p$, we can compute our test statistic $\text{MMD}_b[\mathcal{F}, \mathcal{V}_c, \mathcal{V}_p]$ according to equation 2.22.

We will bootstrap the distribution of MMD_b under the null hypothesis, according to the procedure described in 2.6.3. Comparing our test statistic to the bootstrap estimates, we obtain a p-value.

Following the same procedure, we perform the test *circle vs. square* from the samples of point clouds $\mathcal{T}_c, \mathcal{T}_s$. All tests are performed for a standard deviation $\sigma \in \{2, 4, 6, 8, 10\}$ (for the Gaussian noise when sampling from the shape) and sample size $n \in \{20, 40, 60, 80, 100\}$. For all experiments $N = 10000$ bootstrap samples were used to calculate the p-values.

Results

We summarize the results from the statistical tests for *circle vs. pine* in Table 5.1.1. The results are presented for the different levels of noise (standard deviation σ) and sample size. The statistical test is able to distinguish between samples from probability distributions over tame parametrized vector spaces induced by the different noisy shapes. The statistical test is able to distinguish more clearly over distributions induced by point clouds sampled with less noise. It is also able to distinguish more clearly between distributions when more samples from each of the distributions are used.

Similarly the results for *circle vs. square* are presented in Table 5.1.2. From Figure 5.1.1 we saw that the stable rank between point clouds sampled from the shapes used in this experiment are similar. While this is reflected somewhat in the higher p-values obtained in this test compared to the previous one (for combinations of high noise and small sample size) it is perhaps surprising that the statistical test is able to distinguish between the two samples at all, since the tests are based on topological methods. It appears that topological data analysis (through persistent homology) is much more sensitive than what its topological origins might indicate. Since persistent homology records how topological features (e.g. H_1 homology classes in this case) appear and disappear when varying a space scale it is able to capture much more than the simple fact that both a circle and a square have one hole (of similar dimension: the diameter of the circle is equal to the length of the side of the square). It may for instance detect that the hole for the square disappears later (since the diameter of the point cloud, i.e. the maximum distance between any two points, will be larger). It may also detect differences in noise patterns, e.g. holes with short persistence that are the result of the Gaussian noise and not of the original geometric shape. The shapes might induce different patterns in the noise (for instance, possibly more short persistence holes might form in the corners of the square) which can be picked up by statistical test.

5.1.2 OASIS dataset

Background

In the previous section we experimented on synthetic data. We now wish to instead apply the two-sample hypothesis test to a real dataset. The statistical test will be based

on a dataset from the Open Access Series of Imaging Studies (OASIS) [33] and will follow the exact experimental design of [30], with the only difference being that the kernels used are those described in 3.4 instead of the scale-space kernel (described in 2.7). Besides carrying practical advantages, this approach will allow us to isolate the impact of the kernel on the results of the statistical test.

Data

The OASIS-2 dataset consists of longitudinal MRI data in older adults, some demented and some nondemented. A sample image is shown to the left in Figure 5.1.3. From this longitudinal data we only keep the first session for each individual. We thus end up with 62 observations in the first group and 66 in the second group. The discrimination between nondemented and demented is based on a clinical assessment called Clinical Dementia Rating (CDR). Individuals categorized as demented in this study present very mild or mild levels of dementia based on CDR. Through a process described in the next section, the MRI data for individuals of the two classes is transformed into two samples of tame parametrized vector spaces, on which the two-sample hypothesis test can be applied. The dataset is described in detail in [33].

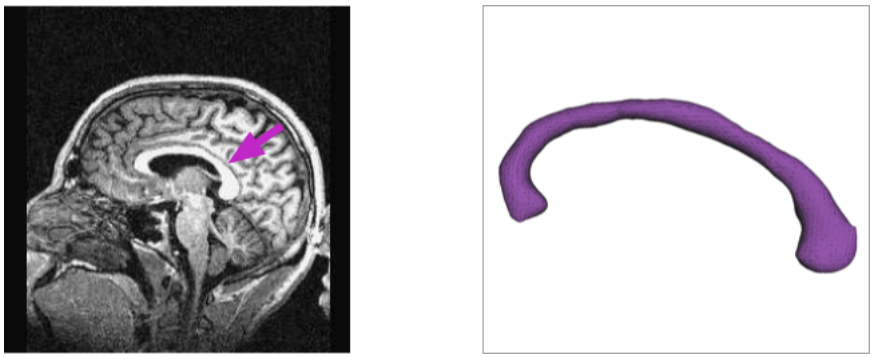


Figure 5.1.3: **Left**: a sample image from the OASIS data set. The arrow points out the corpus callosum region. **Right**: a sample triangle mesh generated from an image in the data set.

Method

From MRI to persistent homology

The processing pipeline of the dataset – from MRI to tame parametrized vector spaces – is quite involved and we refer to [30] for the details. In short a triangle mesh is generated based on the MRI from the segmentation of the corpus callosum, a structure that connects the two hemispheres of the brain. To the right in Figure 5.1.3 an example

of such triangle mesh is shown. This triangle mesh $\mathcal{G}$ does in fact constitute a simplicial complex, that is it is made up of vertices, edges and 2-simplices i.e. the triangles. The heat kernel signature (HKS) [46] associates a real-valued function to the vertices $\mathcal{V}$ of $\mathcal{G}$, i.e. $g : \mathcal{V} \to (\mathbb{R} \to \mathbb{R})$. Inspired by a solution to the heat equation applied in a discrete setting, HKS is a feature descriptor that represents the local and global geometric properties of a vertex. Such descriptors, taken for all the vertices, can be said to capture the features of the shape and are widely used in e.g. shape analysis. Note that the heat kernel signature is used in the processing pipeline and differs from the kernels (scale-space kernel or the kernels developed in this thesis) on which the statistical test is based.

Li et al. [32] observed that this setting is amenable to topological data analysis through persistent homology. We already saw that $\mathcal{G}$ is a simplicial complex. We can then fix the real-valued function attached to each vertex $\mathcal{V}$ at a discretization $t \in \mathbb{R}$ (the whole procedure can subsequentially be repeated for different values of t), giving a map from vertices to a value in $\mathbb{R}$, i.e. $g_t : \mathcal{V} \to \mathbb{R}$. We can now extend the map g_t from the set of vertices to a map defined on any simplex (we defined simplicial complexes and simplices in 2.1.5) by defining $g_t : \sigma \to \mathbb{R}$ as $g_t(\sigma) = \max_{v \in \sigma, v \in \mathcal{V}} g_t(v)$. We thus obtain a real-valued function on simplices of $\mathcal{G}$. We take $\mathcal{G}^\alpha = g_t^{-1}((-\infty, \alpha])$, i.e. in $\mathcal{G}^\alpha$ we only consider the simplices $\{\sigma \in \mathcal{G} \text{ s.t. } g_t(\sigma) \leq \alpha\}$. By varying α from $-\infty$ to $+\infty$ we thus obtain a filtered simplicial complex as described in 2.1.5 of the form $\mathcal{G}^{\alpha_1} \subset \mathcal{G}^{\alpha_2} \subset \mathcal{G}$ for $\alpha_1 \leq \alpha_2$. This process is an example of a sublevel set filtration, which is appropriate when the data can directly be modeled as a simplicial complex together with a real-valued function but not necessarily as a distance space (in which case Vietoris-Rips 2.1.4 will serve as an intermediate step). Finally, H_1 homology is chosen and persistent homology is performed on the filtered simplicial complex thus obtained for each observation in the dataset.

Kernel computation and statistical test

The processing pipeline described above is reproduced using the code provided by the authors of [30]. The Heat Kernel Signature is discretized at $K = 20$ values $t_1, ..., t_K$. Since the sublevel set filtration and further persistent homology is performed based on those 20 sets of vertex values, we end up with 20 tame parametrized vector spaces for each observation in the dataset. They are further grouped into sets of samples for our

two classes $\mathcal{V}_a^{t_k} = \{V_i\}_{i=1}^{62}$ and $\mathcal{V}_b^{t_k} = \{V_i\}_{i=1}^{66}$ where V_i represents the tame parametrized vector space obtained as the result of persistent homology for observation i of the respective class, and for the HKS discretization t_k.

We can then compute the kernel K_C^1 for the standard contour C. We thus end up with a 128×128 Gram matrix G (for the $62 + 66 = 128$ observations in the dataset), i.e. a matrix where $G_{i,j} = K_1^C(V_i, V_j)$. The Gram matrix is normalized according to the following procedure: $G' = \frac{G}{\sqrt{DD^T}}$ where D is the diagonal of G, and the square root and division is taken element-wise (from [19], normalization also performed in [30]). The resulting normalized Gram matrix G' has diagonal entries of value 1.

From this we can compute the test statistic $\text{MMD}_b[\mathcal{F}, \mathcal{V}_a^{t_k}, \mathcal{V}_b^{t_k}]$ according to equation 2.22, same as was done in the previous section on geometric shapes. And finally calculate p-values by bootstrapping the test statistic under the null hypothesis.

Finally we compute the universal kernel K_C^U described in 3.9, also with the standard contour, and repeat the whole procedure with it.

Results

We report the results in Table 5.1.3. We include for comparison the p-values obtained for the scale-space kernel, which we recall is parametrized by σ. In [30] ten different values for the parameter were tested. In the comparison we only include $\sigma = 10^{-3}$ which is the parameter for which the kernel led to the lowest p-value (for any discretization t). The results for the scale-space kernel were collected by running the code provided by the authors.

First, one can note the p-values of some of the tests seem low enough to reject the null hypothesis. However, since multiple tests are done, a p-value correction procedure such as Bonferroni or Benjamini-Hochberg [5] might be desirable, after which (depending on the exact procedure) H_0 may be rejected or not. In this experiment we are however mostly interested in comparing the different kernels. The lower p-values seem to indicate that the kernel based on stable rank is more discriminative than the scale-space kernel in the setting of this experiment. While many more experiments would be needed to judge if this is a general quality of kernels derived from stable rank vs. the scale-space kernel, seeing differences between the two kernels is perhaps not surprising: they are constructed from rather different procedures and hence their discriminative properties can vary, even if it's hard to predict in what direction or give any interpretation. Also, both kernels are stable but with respect to different metrics.

We can also observe, somewhat surprisingly, that lower p-values are obtained using the non-universal kernel (K_C^1), compared to the universal kernel (K_C^U), which has the property that the map from probability distributions to the RKHS corresponding to this kernel is injective. While again no general conclusion can be made based on a single experiment, it may well be that the quality of the embedding or map, that is how rich the geometry in the feature space is and how well it reflects patterns in the space of tame parametrized vector spaces is the most important property to build powerful statistical tests or classification mechanisms able to distinguish between different samples, beyond the theoretical property of universality.

We finally note that experimenting with different contours may have improved the power of the statistical tests, however we did not pursue it here, because optimizing kernels for statistical tests is not obvious. While some approaches for optimizing parametrized kernels are possible such as optimizing among kernels that are linear

	Scale-space	K_C^1	K_C^U
t_1	0.7847	0.9525	0.9523
t_2	0.8375	0.9931	0.9964
t_3	0.9212	0.992	0.9962
t_4	0.9651	0.7606	0.8657
t_5	0.6916	0.312	0.4416
t_6	0.3054	0.1517	0.1837
t_7	0.181	0.0424	0.0503
t_8	0.1723	0.0291	0.0436
t_9	0.2043	0.0412	0.115
t_{10}	0.2396	0.0795	0.2267
t_{11}	0.2521	0.0857	0.2897
t_{12}	0.2712	0.0721	0.2299
t_{13}	0.2867	0.0615	0.2229
t_{14}	0.3019	0.1278	0.3132
t_{15}	0.3401	0.2752	0.5043
t_{16}	0.3981	0.3113	0.5268
t_{17}	0.4822	0.449	0.7546
t_{18}	0.5856	0.2309	0.6185
t_{19}	0.672	0.0633	0.3671
t_{20}	0.7441	0.0751	0.3566

Table 5.1.3: P-values for the scale-space kernel ($\sigma = 10^{-3}$), stable rank K_C^1 and K_C^2 for HKS discretizations $t_1, ..., t_{20}$.

combinations of a set of basis kernels as in [25], optimizing over a large family of contours remains hard in this framework. This is something we instead attempt in the context of machine learning in the next section.

5.2 Machine learning

5.2.1 Point processes

Background

In our first machine learning experiment we want to highlight the importance of the learning algorithm by comparing three learning methods based on the stable rank feature map: the two methods developed in this thesis and a third method introduced in [11].

In 5.1.1 we worked with point clouds sampled with noise from geometric shapes. In this section we will work with another kind of synthetic data: point processes. While point processes are studied in probability theory, they are interesting in the context of

topological data analysis since a realization of a point process is a point cloud (here, in the plane), amenable to analysis through persistent homology using the Vietoris-Rips filtration. Point processes in topological data analysis are studied for example in [11], [6].

Data

In what follows we will work with two types of point processes, constructed as follows:

- Poisson: first the number of points N is sampled from the Poisson distribution parametrized with event rate λ (we will use $\lambda = 200$). Then the N points are sampled from a uniform distribution over the unit square, i.e. $[0,1] \times [0,1]$.
- Matérn: Poisson processes are generated as above but with event rate κ (we will use $\kappa = 40$). Those sampled points form parent points. For each such parent, a number N of child points is sampled from a Poisson distribution with event rate μ (we will use $\mu = 5$). These N points are sampled uniformly from a disk of radius r (here $r = 0.1$) centered at the parent point. Only the child points are part of the point cloud.

In Figure 5.2.1 we show example realizations of the Poisson and the Matérn point processes. We also show the barcode and the stable rank (with standard contour) of the realizations of the point processes, corresponding to the H_1 homology which is what we will work with throughout this section.

We will illustrate machine learning methods based on stable rank for the following binary classification problem. We are given a training set, i.e. a labeled sample of realizations of point processes. The input thus consists of sets of points in $\mathbb{R}^2$ of variable-length. Persistent homology is performed on the point clouds, resulting in a set $\mathcal{T} = \{V_i, y_i\}_{i=1}^N$ where $V_i \in \text{Tame}([0,\infty), \text{Vect})$ representing the H_1 homology and $y_i \in \{0,1\}$ indicates if the point cloud was generated by a Poisson or a Matérn process. Our goal is to device a classification algorithm capable of classifying an out-of-sample observation V^*, derived from a realization of a Poisson or a Matérn point process. Our algorithm is further evaluated based on a test set $\Gamma = \{V_i, y_i\}_{i=1}^M$. In what follows we use perfectly balanced datasets with $N = 140$ and $M = 60$.

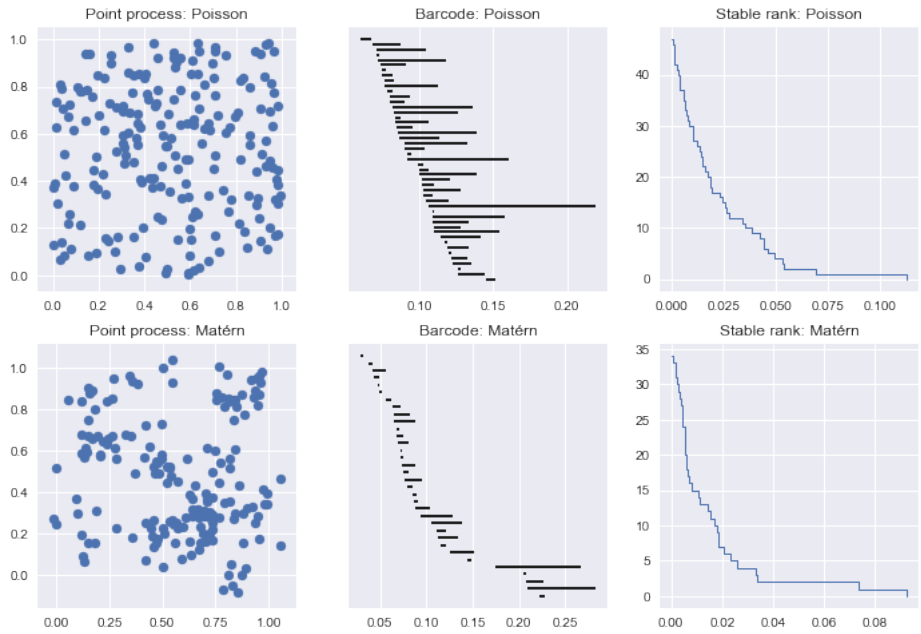


Figure 5.2.1: Two point processes, of type Poisson and Matérn respectively, together with their barcode and stable rank.

Method

Classifying based on distance to mean stable rank

We start by briefly explaining a method from [11] also based on the stable rank feature map. We calculate the mean stable rank g_0 and g_1 for the Poisson and Matérn point process respectively. That is we define $g_k = \frac{1}{N_k} \sum_{(V_i,y_i)\in\mathcal{T}} \widehat{\text{rank}_C(V_i)} I(y_i = k)$ where $k \in \{0,1\}$, I is the indicator function and $N_k = |\{\{V_i, y_i\} \in \mathcal{T} \text{ s.t. } y_i = k\}|$. We choose C to be the standard contour.

We now want to predict the class for a realization of a point process in the test set. To this aim, for each tame parametrized vector space V^* in the test set, we calculate the integral distance to the mean stable rank for the two classes and select the closest class. That is, our decision function is:

$$\arg\min_{y\in\{0,1\}} \int |\widehat{\text{rank}_C(V^*)(t)} - g_y(t)|dt.$$

Neural network

We now build a neural network in PyTorch based on the input layer as described in 4.8. We are able to reach satisfying accuracy after some basic experimentation with the network architecture, which is kept simple for the problem at hand and to avoid

overfitting. We summarize the properties of the network:

- For the input layer we use two parameters μ_1, μ_2, initialized at $\mu_1 = 0.1, \mu_2 = 0.2$ and $\sigma = 0.05$. The number of parameters should be considered a hyperparameter, while their initialization and the value of σ can be heuristically derived from the maximum end point $e_{\max}$ of bars in the barcodes in the training set (or e.g. by looking at Figure 5.2.1), for instance choosing initial values equally spaced between 0 and $e_{\max}$, and σ to some fraction of $e_{\max}$.
- For each μ_i, six features are extracted. That is, we choose $y_1, ..., y_6$ equally spaced between 0 and 100, then the extracted feature corresponds to the y_j:th percentiles. In total, twelve features are thus extracted. Those features can be said to form the unique hidden layer of the neural network. Weights $\eta_1, ..., \eta_{12}$ together with a bias term η_0 are attached to this layer.
- In the output layer, the logistic function transforms the input from the hidden layer into a probability of assigning to the first class. In this sense, the network can be seen as a logistic regression, only instead of a fixed vector as input we have an input layer with trainable parameters.
- The network is trained with subgradient descent for 100 iterations with a learning rate of 0.02. The parameters $\eta_0, ..., \eta_{12}$ are initialized at 0.

In Figure 5.2.2 we show an example run of subgradient descent over a training set, to illustrate that the parameter values adjust to give a lower error rate.

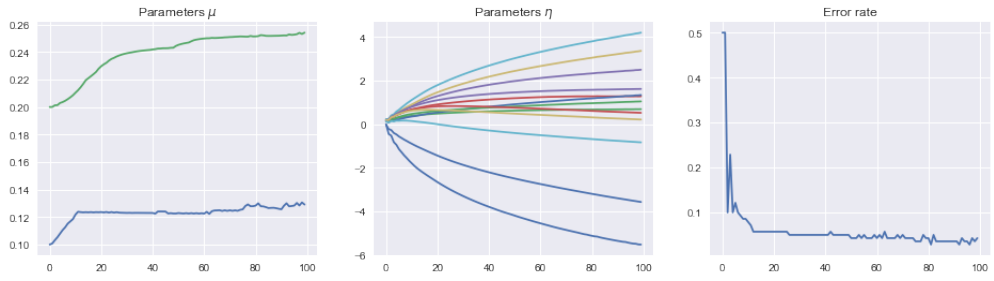


Figure 5.2.2: Values of parameters of the network and error rate on the training set, plotted against the iteration number.

SVM with stable rank kernel

We further apply SVM based on the kernel K_C^1 with standard contour C. To this aim we simply need to construct the Gram matrix for the training set, i.e. the matrix $G^{(1)}$

	Distance to mean	Neural network	SVM
1	6.7	6.7	1.7
2	5	5	8.3
3	10	6.7	5
4	8.3	5	3.3
5	8.3	6.7	5
6	8.3	13.3	6.7
7	15	10	13.3
8	8.3	15	5
9	13.3	10	8.3
10	6.7	8.3	5

Table 5.2.1: Error rate from 10 runs for the three machine learning methods based on stable rank. Blue indicates the lowest error rate for each run.

where $G^{(1)}_{i,j} = K^C_1(V_i, V_j)$, $1 \leq i, j \leq N$, $V_i, V_j \in \mathcal{T}$, as well as the matrix $G^{(2)}$ where $G^{(2)}_{i,j} = K^C_1(V_i, W_j)$ where $1 \leq i \leq N$, $1 \leq j \leq M$, $V_i \in \mathcal{T}, W_j \in \Gamma$, i.e. the interactions between elements the training and the test set. Finally, scikit-learn [39] with the custom kernel option is used to fit the SVM model and produce predictions for the test set, based on those matrices.

Results

We illustrate the error rate from our three machine learning methods based on stable rank in Table 5.2.1. We collect error rate from 10 different runs, i.e. we generate 10 pairs of training and test sets. We see that SVM reports the lowest error rate, followed by the neural network.

5.2.2 Graph classification

Background

While in our previous experiment the goal was to compare different learning methods based on the stable rank feature map on a small synthetic dataset, we will now use a real dataset. The dataset contains more observations, and for each observation the rank of the tame parametrized vector space (which determines the cardinality N of the set $\{(s_i, e_i)\}_{i=1}^N$ resulting from its decomposition, used as input to the learning algorithms) is bigger than in the previous experiment. Therefore, in this section we will only consider neural networks based on the input layer developed in 4, able to handle larger datasets. The dataset considered has been used as a benchmark in various

papers applying very different methods, some related to topological data analysis and some not. This comparison will help shed some light on our method. We will however content ourselves with showing that our method can be applied to this problem and that the performance can approach that of other methods.

Different problems related to learning on graphs – such as predicting missing edges or predicting vertex labels in a graph – are grouped under the name *graph machine learning* [51]. Another such problem, which we will concern ourselves with in this experiment, is graph classification: based on a training set $\{\mathcal{G}_i, y_i\}_{i=1}^N$ where $\mathcal{G}_i = (\mathcal{V}, \mathcal{E})$ is an undirected graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$, and $y_i \in \{1, 2, ..., d\}$ is the class of observation i, the goal is to learn a function taking any graph $\mathcal{G}^*$ to its class y^*. An example of such a problem is to identify whether a given protein is an enzyme [50]. The protein structure is represented as a graph where vertices correspond to atoms and edges correspond to chemical bonds between atoms. The training set is constructed from the graph structure of already known proteins.

Various methods have been applied to the graph classification problem; we will only give a brief overview of them. A first approach is to extract a vector of representative features for each graph, e.g. number of vertices and edges, average degree (number of edges that are incident to a vertex) or more complicated features. Standard machine learning algorithms can then be used, such as Random Forest [3]. A second approach is to develop a kernel capturing the similarity between two graphs [50], to leverage kernel learning methods (presented in 2.8.1). Finally, Graph Convolutional Networks – which generalize Convolutional Neural Networks originally developed for the Euclidean domain – can be effective, particularly on graphs where vertex labels are available [37].

Data

On Reddit, a member starts a discussion (called a *thread*), after which other members can comment or answer to previous comments, in which way a discussion is engaged. Such a thread can be represented by a graph, where each vertex represents a user and an edge represents whether two users interact in the thread e.g. by responding to each other's comments. On Reddit, threads belong to sub-communities, often thematic, called subreddits. One can posit that there is a difference in communication patterns between threads from different subreddits, and that this can be captured by the graph

representation. As an example, for threads in the subreddit ”Ask me anything” there may be discussions mostly with the original poster of the thread, whereas for other subreddits discussions may occur mostly between small groups of commentators. In general the patterns will be subtle, making the classification based solely on the graph representation a challenging task. An example of graph generated from a thread in the dataset is shown in Figure 5.2.3 (for clarity only the main connected component of the graph is shown, image created with Gephi [4]).

The dataset *REDDIT-MULTI-5K* was assembled by the authors of [52] by crawling 1000 top threads from each of the subreddits *worldnews, videos, AdviceAnimals, aww, mildlyinteresting*. The communication graphs are obtained from these threads, resulting in a set of 5000 graphs labeled with the subreddit from which the thread was crawled. On average the graphs in the dataset have 508.5 vertices and 2380 edges.

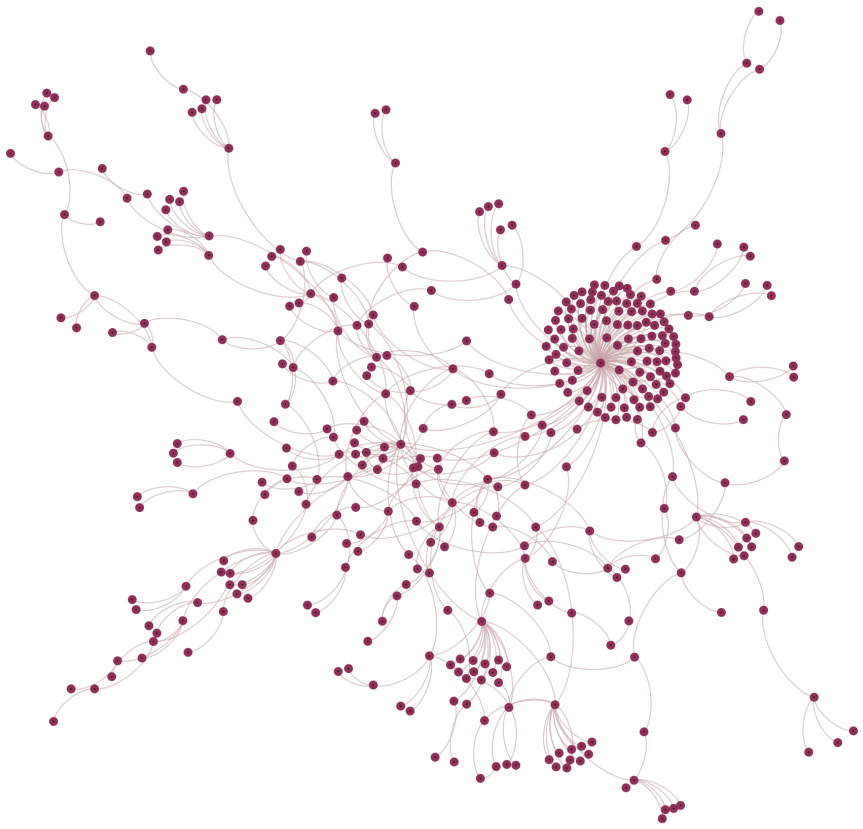


Figure 5.2.3: An example of graph generated from a Reddit thread in the dataset.

Method

Filtration and persistent homology

We have seen that methods such as graph kernels and Graph Convolutional Networks can be used for the graph classification problem. As an alternative, methods derived from topological data analysis can be envisioned. For this to be possible we first need to represent the graphs as filtered simplicial complexes. We will follow the method of [27], which has similarities with the conversion of triangle meshes to simplicial complexes operated in 5.1.2. We observe that a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a simplicial complex made up of vertices and edges but no higher-dimensional simplices. In order to obtain a filtered simplicial complex we start by defining a real-valued function on the vertices: the function $g : \mathcal{V} \to \mathbb{R}$ will map a vertex to its normalized degree, i.e. $g(v) = \frac{\deg(v)}{\max_{v \in \mathcal{V}} \deg(v)}$. We can now extend the map g from the set of vertices to a map defined on any simplex by defining $g : \sigma \to \mathbb{R}$ as $g(\sigma) = \max_{v \in \sigma, v \in \mathcal{V}} g(v)$. We thus obtain a real-valued function on simplices of $\mathcal{G}$ (i.e. both on vertices and on edges). We take $\mathcal{G}^\alpha = g^{-1}((-\infty, \alpha])$, i.e. in $\mathcal{G}^\alpha$ we only consider the simplices $\{\sigma \in \mathcal{G} \text{ s.t. } g(\sigma) \leq \alpha\}$. By varying α from $-\infty$ to $+\infty$ we thus obtain a filtered simplicial complex. We note that the choice of function g will have a large effect on the overall performance. Other choices can be made, sometimes with better results, as in [54] where the Jaccard index is used. In this experiment we will stick with using the vertex degrees since our focus is on the last part of the learning pipeline (from persistent homology to classification results) to help demonstrate the usefulness of the neural network input layer.

Persistent homology is performed on the filtered simplicial complexes. From the definition of n-th homology of simplicial complexes in 2.1.3 we can see that because our simplicial complexes don't contain any simplices of dimension bigger than 1, during decomposition all H_1 topological features will be essential, that is they will persist forever. We also note that for a graph (to the contrary of simplicial complexes obtained as the result of e.g. the Vietoris-Rips filtration) there may be multiple essential H_0 features, corresponding to the different connected components of the graph. Such essential features are in fact easier to deal with than non-essential features in a machine learning problem. Indeed essential features $\{(s_i, \infty)\}_{i=1}^n$ may be seen as a variable-length set of scalars $\{s_i\}_{i=1}^n$ on which for instance the methods of machine learning on sets reviewed in 2.8.3 may be applied. Even though essential features have an impact on the accuracy of a graph classification algorithm as shown in [27] we choose

to disregard them in this experiment, as the goal is simply to try out the neural network input layer (which is designed for the problem of handling non-essential features) on a real dataset.

To give an overview of the processed dataset we stack all the non-essential features of all the observations in each class. We show the diagrams for three of the five classes in the classification problem in Figure 5.2.4.

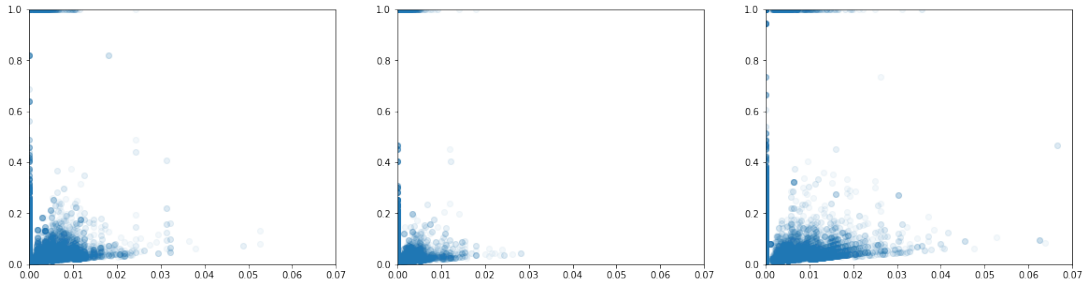


Figure 5.2.4: Stacked persistence diagrams for three of the five classes in the graph classification problem.

Network architecture

A simple architecture is used for which we summarize the properties:

- A single parameter μ is used, initialized at $\mu = 0.1$. The scale parameter is chosen at $\sigma = 0.03$.
- 20 features are extracted. That is, we choose $y_1, ..., y_{20}$ equally spaced between 0 and 100. Batch normalization and ReLU activation is applied to this output. This forms the first hidden layer, which is connected to a second hidden layer with 10 units. This layer is further connected to the output layer.
- Cross-entropy loss is used. The network is optimized with the Adam optimizer with a fixed learning rate of 0.01, and batch size 128, for 100 epochs.

Results

10-fold validation is performed, resulting in accuracy of 45.69% and a standard deviation of 2.17%. We recall that since we have a 5-class classification problem, an algorithm guessing at random would achieve an accuracy of 20%. Thus the neural network is clearly learning something. In Table 5.2.2 we show accuracies achieved by other methods as reported in [27]. Methods derived from topological data analysis

	Accuracy
Non-topological methods	
Graph kernel [50]	41.0
Deep graph kernel [52]	41.3
PSCN [37]	49.1
Random Forest [3]	50.9
Topological methods	
Topological signatures [27] w/o essential	49.1
Topological signatures [27] w/ essential	54.5
Method in this thesis (w/o essential)	45.7

Table 5.2.2: Accuracies for various graph classification methods on the Reddit dataset.

generally perform well on this type of problem, which is also the case of our method. The method closest to ours [27], using the same graph filtration and also only using non-essential features, achieves a somewhat higher accuracy at 49.1%. It is unclear whether those last percentage points in accuracy could be achieved by tuning the architecture and hyperparameters of the neural network or if the reasons are more inherent to the input layer.

A surprising fact observed while performing this experiment was that, to the contrary of the experiment on point processes, adding more parameters μ to the input layer did not seem to help increase accuracy for this problem. In fact, during the training process, multiple parameters often converged to similar values. However a look at the processed dataset (e.g. the persistence diagrams) and Figure 5.2.4 might help explain the situation. In fact a majority of points start at 0, and the few that don't start very close to 0 (note that the axes are not equal in the diagrams). It might thus be that the set of non-essential features could be represented as a set of scalars $\{e_i\}_{i=1}^{N}$ without much loss of information. In such case the value added of several parametrization with different μ would be null.

Chapter 6

Conclusion and future work

In this thesis we presented kernels, a statistical test and learning methods based on the stable rank, with the goal of concretely advancing the understanding and usefulness of this invariant, and more generally distance-based invariants in topological data analysis. We tried to keep some unifying themes throughout the thesis:

- The proposed methods illustrate some of the recent research areas in the intersection of topological data analysis on the one hand, and statistics and machine learning on the other.
- We highlighted the importance of the choice of the feature map and discussed its properties, both theoretical, such as stability, and empirical. All proposed methods were based on the stable rank invariant, thus showing how a single class of feature maps can give rise to a wide variety of statistical methods. Each element of this class is defined by a notion of distance between topological summaries of the data. One of the methods illustrated how the optimal notion of distance could be learnt for a particular machine learning problem at hand.
- We illustrated some commonalities between classical statistical methods and machine learning, e.g. classification methods and two-sample hypothesis tests. We further showed that they can be developed based partly on the same theoretical building blocks (in our case, of kernels).

In this process, many areas were of course left unexplored, and we can mention a few of them on the theoretical level. First, it is possible that the construction of a universal kernel could have been more direct, or that our second kernel is in fact already

universal. Second, while we argued that the assumptions on parametrized vector spaces (obtained during the computation of persistent homology) made to develop the kernels are not limiting in practice, a more in-depth analysis as to what assumptions are necessary for each property of the kernels would be interesting. Finally, it would be interesting to explore if the neural network input layer or another construction derived from the stable rank invariant could be put to use in the construction of a topological loss function in the context of generative models or autoencoders, which requires backpropagation to the data space or latent space.

Bibliography

[1] Adams, Henry, Emerson, Tegan, Kirby, Michael, Neville, Rachel, Peterson, Chris, Shipman, Patrick, Chepushtanova, Sofya, Hanson, Eric, Motta, Francis, and Ziegelmeier, Lori. "Persistence images: A stable vector representation of persistent homology". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 218–252.

[2] Arcones, Miguel A and Gine, Evarist. "On the bootstrap of U and V statistics". In: *The Annals of Statistics* (1992), pp. 655–674.

[3] Barnett, Ian, Malik, Nishant, Kuijjer, Marieke L, Mucha, Peter J, and Onnela, Jukka-Pekka. "Feature-based classification of networks". In: *arXiv preprint arXiv:1610.05868* (2016).

[4] Bastian, Mathieu, Heymann, Sebastien, and Jacomy, Mathieu. "Gephi: an open source software for exploring and manipulating networks". In: *Third international AAAI conference on weblogs and social media*. 2009.

[5] Benjamini, Yoav and Hochberg, Yosef. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the Royal statistical society: series B (Methodological)* 57.1 (1995), pp. 289–300.

[6] Biscio, Christophe AN and Møller, Jesper. "The accumulated persistence function, a new useful functional summary statistic for topological data analysis, with a view to brain artery trees and spatial point process applications". In: *Journal of Computational and Graphical Statistics* 28.3 (2019), pp. 671–681.

[7] Brüel-Gabrielsson, Rickard, Nelson, Bradley J, Dwaraknath, Anjan, Skraba, Primoz, Guibas, Leonidas J, and Carlsson, Gunnar. "A topology layer for machine learning". In: *arXiv preprint arXiv:1905.12200* (2019).

[8] Bubenik, Peter. “Statistical topological data analysis using persistence landscapes”. In: *The Journal of Machine Learning Research* 16.1 (2015), pp. 77–102.

[9] Carrière, Mathieu, Chazal, Frédéric, Datashape, Inria Saclay, Ike, Yuichi, Lacombe, Théo, Royer, Martin, and Umeda, Yuhei. “PersLay: A Simple and Versatile Neural Network Layer for Persistence Diagrams”. In: *stat* 1050 (2019), p. 5.

[10] Casella, George and Berger, Roger L. *Statistical inference*. Vol. 2. Duxbury Pacific Grove, CA, 2002.

[11] Chachólski, Wojciech and Riihimäki, Henri. “Metrics and stabilization in one parameter persistence”. In: *SIAM Journal on Applied Algebra and Geometry* 4.1 (2020), pp. 69–98.

[12] Chazal, Frédéric, Cohen-Steiner, David, Glisse, Marc, Guibas, Leonidas J, and Oudot, Steve Y. “Proximity of persistence modules and their diagrams”. In: *Proceedings of the twenty-fifth annual symposium on Computational geometry*. 2009, pp. 237–246.

[13] Chazal, Frédéric, Fasy, Brittany, Lecci, Fabrizio, Michel, Bertrand, Rinaldo, Alessandro, and Wasserman, Larry. “Subsampling methods for persistent homology”. In: *International Conference on Machine Learning*. 2015, pp. 2143–2151.

[14] Christmann, Andreas and Steinwart, Ingo. “Universal kernels on non-standard input spaces”. In: *Advances in neural information processing systems*. 2010, pp. 406–414.

[15] Clough, James R, Oksuz, Ilkay, Byrne, Nicholas, Zimmer, Veronika A, Schnabel, Julia A, and King, Andrew P. “A Topological Loss Function for Deep-Learning based Image Segmentation using Persistent Homology”. In: *arXiv preprint arXiv:1910.01877* (2019).

[16] Dindin, Meryll, Umeda, Yuhei, and Chazal, Frédéric. “Topological Data Analysis for Arrhythmia Detection through Modular Neural Networks”. In: *arXiv preprint arXiv:1906.05795* (2019).

[17] Dudley, Richard M. *Real analysis and probability*. Chapman and Hall/CRC, 2018.

[18] Efron, Bradley and Tibshirani, Robert J. *An introduction to the bootstrap*. CRC press, 1994.

[19] Feragen, Aasa, Kasenburg, Niklas, Petersen, Jens, Bruijne, Marleen de, and Borgwardt, Karsten. "Scalable kernels for graphs with continuous attributes". In: *Advances in neural information processing systems*. 2013, pp. 216–224.

[20] Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[21] Gäfvert, Oliver. "Topology-based metric learning". Poster presented at the conference TAGS - Linking Topology to Algebraic Geometry and Statistics, Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany.

[22] Gidea, Marian and Katz, Yuri. "Topological data analysis of financial time series: Landscapes of crashes". In: *Physica A: Statistical Mechanics and its Applications* 491 (2018), pp. 820–834.

[23] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT press, 2016.

[24] Gretton, Arthur, Borgwardt, Karsten M, Rasch, Malte J, Schölkopf, Bernhard, and Smola, Alexander. "A kernel two-sample test". In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773.

[25] Gretton, Arthur, Sejdinovic, Dino, Strathmann, Heiko, Balakrishnan, Sivaraman, Pontil, Massimiliano, Fukumizu, Kenji, and Sriperumbudur, Bharath K. "Optimal kernel choice for large-scale two-sample tests". In: *Advances in neural information processing systems*. 2012, pp. 1205–1213.

[26] Hofer, Christoph D, Kwitt, Roland, and Niethammer, Marc. "Learning representations of persistence barcodes". In: *Journal of Machine Learning Research* 20.126 (2019), pp. 1–45.

[27] Hofer, Christoph, Kwitt, Roland, Niethammer, Marc, and Uhl, Andreas. "Deep learning with topological signatures". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1634–1644.

[28] Iijima, Taizo. "Basic theory on the normalization of pattern (in case of typical one-dimensional pattern)". In: *Bulletin of Electro-technical Laboratory* 26 (1962), pp. 368–388.

[29] Kovacev-Nikolic, Violeta, Bubenik, Peter, Nikolić, Dragan, and Heo, Giseon. "Using persistent homology and dynamical distances to analyze protein binding". In: *Statistical applications in genetics and molecular biology* 15.1 (2016), pp. 19–38.

[30] Kwitt, Roland, Huber, Stefan, Niethammer, Marc, Lin, Weili, and Bauer, Ulrich. "Statistical topological data analysis-a kernel perspective". In: *Advances in neural information processing systems*. 2015, pp. 3070–3078.

[31] Lesnick, Michael. "The theory of the interleaving distance on multidimensional persistence modules". In: *Foundations of Computational Mathematics* 15.3 (2015), pp. 613–650.

[32] Li, Chunyuan, Ovsjanikov, Maks, and Chazal, Frederic. "Persistence-based structural recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1995–2002.

[33] Marcus, Daniel S, Fotenos, Anthony F, Csernansky, John G, Morris, John C, and Buckner, Randy L. "Open access series of imaging studies: longitudinal MRI data in nondemented and demented older adults". In: *Journal of cognitive neuroscience* 22.12 (2010), pp. 2677–2684.

[34] Mileyko, Yuriy, Mukherjee, Sayan, and Harer, John. "Probability measures on the space of persistence diagrams". In: *Inverse Problems* 27.12 (2011), p. 124007.

[35] Moor, Michael, Horn, Max, Rieck, Bastian, and Borgwardt, Karsten. "Topological Autoencoders". In: *arXiv preprint arXiv:1906.00722* (2019).

[36] Nielson, Jessica L, Paquette, Jesse, Liu, Aiwen W, Guandique, Cristian F, Tovar, C Amy, Inoue, Tomoo, Irvine, Karen-Amanda, Gensel, John C, Kloke, Jennifer, Petrossian, Tanya C, et al. "Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury". In: *Nature communications* 6 (2015), p. 8581.

[37] Niepert, Mathias, Ahmed, Mohamed, and Kutzkov, Konstantin. "Learning convolutional neural networks for graphs". In: *International conference on machine learning*. 2016, pp. 2014–2023.

[38] Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca, and Lerer, Adam. "Automatic differentiation in pytorch". In: (2017).

[39] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[40] Perea, Jose A, Munch, Elizabeth, and Khasawneh, Firas A. "Approximating continuous functions on persistence diagrams using template functions". In: *arXiv preprint arXiv:1902.07190* (2019).

[41] Rasmussen, Carl Edward. "Gaussian processes in machine learning". In: *Summer School on Machine Learning*. Springer. 2003, pp. 63–71.

[42] Reininghaus, Jan, Huber, Stefan, Bauer, Ulrich, and Kwitt, Roland. "A stable multi-scale kernel for topological machine learning". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 4741–4748.

[43] Scolamiero, Martina, Chachólski, Wojciech, Lundman, Anders, Ramanujam, Ryan, and Öberg, Sebastian. "Multidimensional persistence and noise". In: *Foundations of Computational Mathematics* 17.6 (2017), pp. 1367–1406.

[44] Serfling, Robert J. *Approximation theorems of mathematical statistics*. Vol. 162. John Wiley & Sons, 2009.

[45] Sriperumbudur, Bharath K, Gretton, Arthur, Fukumizu, Kenji, Schölkopf, Bernhard, and Lanckriet, Gert RG. "Hilbert space embeddings and metrics on probability measures". In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1517–1561.

[46] Sun, Jian, Ovsjanikov, Maks, and Guibas, Leonidas. "A concise and provably informative multi-scale signature based on heat diffusion". In: *Computer graphics forum*. Vol. 28. 5. Wiley Online Library. 2009, pp. 1383–1392.

[47] Tralie, Christopher, Saul, Nathaniel, and Bar-On, Rann. "Ripser.py: A Lean Persistent Homology Library for Python". In: *The Journal of Open Source Software* 3.29 (Sept. 2018), p. 925. DOI: 10.21105/joss.00925. URL: https://doi.org/10.21105/joss.00925.

[48] Turner, Katharine, Mileyko, Yuriy, Mukherjee, Sayan, and Harer, John. "Fréchet means for distributions of persistence diagrams". In: *Discrete & Computational Geometry* 52.1 (2014), pp. 44–70.

[49] Umeda, Yuhei. “Time series classification via topological data analysis”. In: *Information and Media Technologies* 12 (2017), pp. 228–239.

[50] Vishwanathan, S Vichy N, Schraudolph, Nicol N, Kondor, Risi, and Borgwardt, Karsten M. “Graph kernels”. In: *Journal of Machine Learning Research* 11.Apr (2010), pp. 1201–1242.

[51] Wu, Zonghan, Pan, Shirui, Chen, Fengwen, Long, Guodong, Zhang, Chengqi, and Philip, S Yu. “A comprehensive survey on graph neural networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[52] Yanardag, Pinar and Vishwanathan, SVN. “Deep graph kernels”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 1365–1374.

[53] Zaheer, Manzil, Kottur, Satwik, Ravanbakhsh, Siamak, Poczos, Barnabas, Salakhutdinov, Russ R, and Smola, Alexander J. “Deep sets”. In: *Advances in neural information processing systems*. 2017, pp. 3391–3401.

[54] Zhao, Qi and Wang, Yusu. “Learning metrics for persistence-based summaries and applications for graph classification”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 9855–9866.

[55] Zomorodian, Afra and Carlsson, Gunnar. “Computing persistent homology”. In: *Discrete & Computational Geometry* 33.2 (2005), pp. 249–274.

TRITA SCI-GRU 2020:056