Journal of Machine Learning Research 20 (2019) 1-45 Submitted 6/16; Revised 2/18; Published 2/19

Kernels for Sequentially Ordered Data

Franz J. Király F.KIRALY@UCL.AC.UK
Department of Statistical Science
University College London
London WC1E 6BT, United Kingdom

Harald Oberhauser OBERHAUSER@MATHS.OX.AC.UK
Mathematical Institute
University of Oxford
Oxford OX2 6GG, United Kingdom

Editor: Le Song

Abstract

We present a novel framework for learning with sequential data of any kind, such as multivariate time series, strings, or sequences of graphs. The main result is a "sequentialization" that transforms any kernel on a given domain into a kernel for sequences in that domain. This procedure preserves properties such as positive definiteness, the associated kernel feature map is an ordered variant of sample (cross-)moments, and this sequentialized kernel is consistent in the sense that it converges to a kernel for paths if sequences converge to paths (by discretization). Further, classical kernels for sequences arise as special cases of this method. We use dynamic programming and low-rank techniques for tensors to provide efficient algorithms to compute this sequentialized kernel.

Keywords: Sequential data, kernels, signature, ordered moments, signature kernels

1. Introduction

Sequentially ordered data are ubiquitous in modern science, occurring as time series, location series, or, more generally, sequentially observed samples of structured objects. Two stylised facts make learning with sequential data an ongoing challenge:

(A) Sequential data is very diverse, e.g. it includes sequences of different length of scalars, letters, images, graphs (in time series, text, video, network evolution) or even heterogeneous combinations of these. This diversity is usually addressed by *ad-hoc approaches, extensive pre-processing and manual extraction of features*, thus specific to the data at hand.

(B) Sequential data is often large, with sequences easily obtaining the length of hundreds, thousands, millions. Hence, the data sets quickly become *very huge*, and with them computational cost.

In this paper, we present a novel approach to learning with sequential data based on joining ideas from *stochastic analysis* and *kernel learning*, addressing the points above:

(a) The (discretized) *signature* is a *canonical feature map* for sequences. It can intuitively be described as an ordered version of sample moments. It completely describes a sequence and makes sequences of different size and length comparable.

(b) The *kernel trick* applied to signature features avoids the combinatorial explosion of coordinates inherent to the signature feature map (in analogy to the classic polynomial kernel in $\mathbb{R}^d$).

The main results of this article can be described as follows: if $\mathcal{X}$ is a topological space ("structured objects") and we denote with $\mathcal{X}^+ = \bigcup_{\ell \geq 1} \mathcal{X}^\ell$ the set of arbitrary length sequences in $\mathcal{X}$ ("sequences of structured objects") then we construct a map that takes

$$\text{a kernel } \mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \text{ to a kernel } \mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}.$$

We call k^+ the *discretized signature kernel* (or *sequentialization*) over k and provide efficient algorithms to evaluate k^+. This transformation is canonical in the sense that it preserves important properties of k, such as positive definiteness and it is consistent in the sense that $\mathrm{k}(x, y)$ for $x, y \in \mathcal{X}^+$ converges to an inner product of a "feature map" for paths that is classic in stochastic analysis (the path signature) if the sequences x, y converge to paths (from discrete to continuous time).

Classical kernels for sequences such as string, global alignment or relation-convolution kernels can be identified as special cases of the discretized signature kernel k^+; for example string kernels arise by taking $\mathcal{X}$ to be a finite set of "letters". More interestingly, even for $\mathcal{X} = \mathbb{R}$ this yields new kernels for time series by the sequentialization of classic non-linear kernels such as Gaussian, Laplace, etc. From a methodological point of view, this gives a canonical way to transform static features (resp. kernels k) for structured objects into features (resp. kernels k^+) for evolving structured objects.

1.1. Kernels for paths

We first discuss the situation for path-valued observations (continuous time) rather than sequence-valued observations (discrete time) — the latter then follows by discretizing time. That is, if we denote with $\mathcal{P}_\mathcal{X} \subset \mathcal{X}^{[0,1]}$ a sufficiently regular subset of the space of paths that evolve in $\mathcal{X}$ (without loss of generality the paths are parameterized by $[0, 1]$), our aim is to transform

$$\text{a kernel } \mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \text{ into a kernel } \mathrm{k}^\oplus : \mathcal{P}_\mathcal{X} \times \mathcal{P}_\mathcal{X} \to \mathbb{R}.$$

Therefore, first recall that $\mathcal{X} \ni u \mapsto \mathrm{k}_u := \mathrm{k}(u, \cdot) \in \mathcal{H} \subset \mathbb{R}^\mathcal{X}$ can be thought of as the "(canonical) feature map" induced by k. It is then natural to proceed as follows:

1. **Lift a path evolving in data space $\mathcal{X}$ to a path evolving in feature space $\mathcal{H}$.** We lift the path x that evolves in data space $\mathcal{X}$ to a path[1] k_x that evolves in the feature space $\mathcal{H}$ via the kernel k. That is, we map

$$x = (x(t))_{t\in[0,1]} \in \mathcal{P}_\mathcal{X} \text{ to } \mathrm{k}_x = \left(\mathrm{k}_{x(t)}\right)_{t\in[0,1]} \in \mathcal{P}_\mathcal{H}$$

where $\mathcal{P}_\mathcal{H}$ denotes a sufficiently regular subset of $\mathcal{H}^{[0,1]}$.

1. Note the slight abuse of notation that $\mathrm{k}_u \in \mathcal{H}$ if $u \in \mathcal{X}$ and $\mathrm{k}_x \in \mathcal{P}_\mathcal{H}$ if $x \in \mathcal{P}_\mathcal{X}$.

2. **Signature features for paths in $\mathcal{H}$.** We use a feature map that is well-known in stochastic analysis for paths that evolve in *linear* spaces such as $\mathcal{H}$. This map is called the signature map S. It maps a path $h \in \mathcal{P}_\mathcal{H}$ into a series of tensors,

$$h \mapsto \mathrm{S}(h) \in \prod_{m \geq 0} \mathcal{H}^{\otimes m}.$$

   By applying this map to the lifted path $\mathrm{k}_x \in \mathcal{P}_\mathcal{H}$, we get the feature map $\mathcal{P}_\mathcal{X} \to \prod_{m\geq 0} \mathcal{H}^{\otimes m}$, defined as $x \mapsto \mathrm{S}(\mathrm{k}_x)$.

3. **Signature kernels**. Taking inner products yields the kernel

$$\mathrm{k}^{\oplus} : \mathcal{P}_\mathcal{X} \times \mathcal{P}_\mathcal{X} \to \mathbb{R} \text{ defined as inner product } \mathrm{k}^{\oplus}(x, y) = \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y)\rangle.$$

   It turns out, that $\mathrm{k}^{\oplus}$ has a recursive structure that allows for an efficient and robust calculation even though $\mathrm{k}_x, \mathrm{k}_y$ are paths evolving in the (in general infinite-dimensional) state space $\mathcal{H}$.

Below we give more details about the last two points of this construction.

1.2. The signature map

The signature S maps a (sufficiently regular) path $h \in \mathcal{P}_\mathcal{H} \subset \mathcal{H}^{[0,1]}$ into a series of tensors,

$$h \mapsto \mathrm{S}(h) = (\mathrm{S}_m(h))_{m\geq 0} \in \prod_{m\geq 0} \mathcal{H}^{\otimes m}$$

where by convention $\mathcal{H}^{\otimes 0} := \mathbb{R}$ and $\mathrm{S}_0(h) := 1$. The remaining terms $\mathrm{S}_m(h)$ are defined as follows: the first degree, $\mathrm{S}_1(h)$, of the signature is simply the average change,

$$\mathrm{S}_1(h) := \mathbb{E}_t[\dot{h}(t)] \in \mathcal{H},$$

where $\dot{h}$ denotes the derivative of h and the expectation is taken over t sampled uniformly from $[0, 1]$. The second degree of the signature, $\mathrm{S}_2(h)$, is the (non-centred) covariance of changes at two subsequent time points, that is

$$\mathrm{S}_2(h) := \frac{1}{2}\mathbb{E}_{t_1<t_2}[\dot{h}(t_1) \otimes \dot{h}(t_2)] \in \mathcal{H}^{\otimes 2},$$

where the expectation is over the time points t_1, t_2 sampled from the uniform distribution on $[0, 1]$ and put in chronological order. In general, the m-th degree of the signature, $\mathrm{S}_m(h)$, is defined as the m-th moment tensor of the infinitesimal changes, where the expectation is taken over m time points, $t_1, \ldots, t_m$, that are sampled from the order statistic[2] of the uniform distribution on $[0, 1]$, that is

$$\mathrm{S}_m(h) := \frac{1}{m!}\mathbb{E}_{t_1<\cdots<t_m}[\dot{h}(t_1) \otimes \cdots \dot{h}(t_m)] \in \mathcal{H}^{\otimes m}.$$

2. The order statistic for the uniform distribution $[0, 1]$ of m points has a density $p(t_1, \ldots, t_m) = m!1_{t_1<\cdots t_m<1}$

Hence, the signature $\mathrm{S}(h)$ of a path h characterises the sequential structure in h by quantifying dependencies in their change, similar to classic sample moments in $\mathbb{R}^d$. In fact, signatures are in close mathematical analogy to polynomials: the feature map $x \mapsto \mathrm{S}(h)$ is injective and if $h \mapsto f(h)$ is a *non-linear function on pathspace*, then it can be well approximated as a *linear function* of the signature features $\mathrm{S}(h)$, that is $f(h) \approx \langle \ell, \mathrm{S}(h) \rangle$. However, signature features have the following drawbacks

- Even if $\mathcal{H} = \mathbb{R}^d$ is finite dimensional, we have $\mathrm{S}_m(h) \in \left(\mathbb{R}^d\right)^{\otimes m}$, thus the number of (scalar) coordinates needed to store $\mathrm{S}_m(h)$ grows as $O(d^m)$ as m increases. This combinatorial explosion makes the use of signature features costly for paths that evolve in high-dimensional state spaces $\mathcal{H}$ and infeasible for paths evolving in infinite dimensional state spaces $\mathcal{H}$.
- Signatures are only defined for paths evolving in linear spaces; $\mathrm{S}(h)$ does not make sense when h evolves in a general topological space $\mathcal{X}$ since $\dot{h}(t)dt = dh(t)$ is not well-defined.

However, as we will show, both of these shortcomings can be addressed by kernelization.

1.3. Inner products of signature features

The space $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$ is a linear space and by restricting to a subspace, we can work with an inner product space (in fact, again a Hilbert space after completion); that is we define for $(s_m), (t_m) \in \prod_{m\geq 0} \mathcal{H}^{\otimes m}$

$$\begin{aligned}(s_0, s_1, \ldots) + (t_0, t_1, \ldots) &:= (s_0 + t_0, s_1 + t_1, \ldots)\,,\\ \langle (s_0, s_1, \ldots), (t_0, t_1, \ldots) \rangle &:= \sum_{m\geq 0} \langle s_m, t_m \rangle_{\mathcal{H}^{\otimes m}}\end{aligned}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}^{\otimes m}}$ is given as $\langle u_1 \otimes \cdots \otimes u_m, v_1 \otimes \cdots \otimes v_m \rangle_{\mathcal{H}^{\otimes m}} := \prod_{i=1}^m \langle u_i, v_i \rangle_{\mathcal{H}}$. The inner product $\langle \mathrm{S}(g), \mathrm{S}(h) \rangle$ of the signature features of (sufficiently regular) paths $g, h \in \mathcal{P}_{\mathcal{H}}$ is finite and using conditional expectations we get that $\langle \mathrm{S}(g), \mathrm{S}(h) \rangle$ equals

$$\begin{aligned}&\sum_{m\geq 0} \frac{1}{m!^2} \mathbb{E}_{\substack{s_1 < \cdots < s_m \\ t_1 < \cdots < t_m}} [\langle \dot{g}(s_1) \otimes \cdots \otimes \dot{g}(s_m), \dot{h}(t_1) \otimes \cdots \otimes \dot{h}(t_m) \rangle_{\mathcal{H}^{\otimes m}}]\\ =&\sum_{m\geq 0} \frac{1}{m!^2} \mathbb{E}_{\substack{s_1 < \cdots < s_m \\ t_1 < \cdots < t_m}} [\langle \dot{g}(s_1), \dot{h}(t_1) \rangle_{\mathcal{H}} \cdots \langle \dot{g}(s_m), \dot{h}(t_m) \rangle_{\mathcal{H}}]\\ =&\left(1 + \mathbb{E}_{s_1,t_1}\left[\langle \dot{g}(s_1), \dot{h}(t_1) \rangle \cdot \left(1 + \frac{1}{2^2} \mathbb{E}_{\substack{s_2|s_1 \\ t_2|t_1}}\left[\langle \dot{g}(s_2), \dot{h}(t_2) \rangle \cdot \left(1 + \frac{1}{3^2} \mathbb{E}_{\substack{s_3|s_2 \\ t_3|t_2}}[\ldots]\right)\right]\right)\right]\right).\end{aligned}$$

Now applied to $g = \mathrm{k}_x \in \mathcal{P}_{\mathcal{H}}$, $h = \mathrm{k}_y \in \mathcal{P}_{\mathcal{H}}$, using the reproducing property of k we get

$$\langle \dot{\mathrm{k}}_{x(s)}, \dot{\mathrm{k}}_{y(t)} \rangle_{\mathcal{H}} \,\mathrm{d}s\,\mathrm{d}t = \langle \mathrm{d}g(s), \mathrm{d}h(s) \rangle_{\mathcal{H}} = \mathrm{d}_s\,\mathrm{d}_t\,\mathrm{k}(x(s), y(t)) = \kappa(\mathrm{d}s, \mathrm{d}t)$$

where κ denotes the measure on $[0,1]^2$ defined as

$$\kappa([s,t] \times [u,v]) = \mathrm{k}(x(t), y(v)) - \mathrm{k}(x(s), y(v)) - \mathrm{k}(x(t), y(u)) + \mathrm{k}(x(s), y(u)).$$

(Note that this measure does not use differentiability of x, y). Replacing above expectations over the order statistic by explicit integration over $[0,1]$, we arrive at

$$\mathrm{k}^{\oplus}(x,y) = 1 + \int_{s_1,t_1} \left[\kappa(\mathrm{d}s_1, \mathrm{d}t_1) \cdot \left(1 + \int_{\substack{s_2>s_1 \\ t_2>t_1}} \left[\kappa(\mathrm{d}s_2, \mathrm{d}t_2) \cdot \left(1 + \int_{\substack{s_3>s_2 \\ t_3>t_2}} [\dots] \right) \right] \right) \right] . \quad (1)$$

Finally, the data we have access is to is only a sequence $(x(t_1), \dots, x(t_n)) \in \mathcal{X}^+$ rather than the whole path $x = (x(t))_{t\in[0,1]} \in \mathcal{P}_\mathcal{X}$, but replacing the integrals by sums immediately gives an explicit recursive formula for a kernel $\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$ that is determined by a Horner-type[3] recursive evaluation; especially, one never computes the signature map of a path that evolves in the high/infinite-dimensional Hilber space $\mathcal{H}$ directly. This yields a robust and computationally effective formula for a kernel k^+ for sequences, that has well-defined behaviour in the scaling limit when sequences approximate paths. We make the above informal derivation rigorous in the following sections, study its properties and extended it to non-smooth paths in Appendix B.

1.4. Prior art and literature review

Learning with sequential data is a vast field. Prior art that inspired our present approach can be found in

(i) dynamic programming algorithms for sequence comparison in the engineering community,

(ii) kernel and Gaussian processes for sequences in the machine learning community,

(iii) Rough paths in the stochastic process community.

Beyond the above, we are not aware of literature in statistics that deals with sequence-valued data points in a way other than first identifying one-dimensional sequences with real vectors of same size, or even forgetting the sequence structure entirely and replacing the sequences with (order-agnostic) aggregates such as cumulants, quantiles or principal component scores. Below we discuss above three points in more detail:

(i) Dynamic programming algorithms for sequence comparison.

The earliest use of order information in sequences for learning can probably be found in Sakoe and Chiba (1970) and Sakoe (1979) by using editing or distortion distances. These distances are then employed for classification by maximum similarity/minimum distance principles. Through theoretical appeal and efficient computability, sequence comparison methods, later synonymously called dynamic time warping methods, have become one of the standard methods in comparing sequential data Kruskal (1983); Giorgino (2009). Sequence comparison methods in their original formulation can only be directly applied to

3. Horner's scheme to evaluate a uni-variate polynomial $p(x) = \sum_{i=0}^{m} a_i x^i$ is to write $p(x) = a_0 + x\,(a_1 + x\,(a_2 + \cdots + x\,(a_{m-1} + x a_m)))$ and compute the brackets from the inside to the outside. This needs only n additions and n multiplications which is optimal and in contrast to the n additions and $\frac{n^2+n}{2}$ multiplications needed for the naive evaluation of $p(x)$; further, it is numerically more stable since one never adds floats that live on different scales (such as $a_1 x$ and $a_m x^m$). This structural similarity with formula (1) will be the key to our algorithms.

relatively simple distance-based learning algorithms. This has been addressed by combining such methods with kernel learning (discussed below).

(ii) Kernels and Gaussian processes for sequences.

Kernel learning is one of the most popular approaches to make non-linear data of arbitrary kind amenable to classic and scalable linear algorithms and it provides learning theoretical guarantees, see Schölkopf and Smola (2002); Shawe-Taylor and Cristianini (2004); Cucker and Smale (2002). Kernels for strings, that is, sequences of symbols, were among the first to be considered by Haussler (1999) and fast algorithms were obtained a few years later Lohdi et al. (2002); Leslie and Kuang (2004). Kernels based on the above-mentioned dynamic programming (time warping) approach were developed, for sequences of arbitrary objects leading to so-called alignment kernel, see Bahlmann et al. (2002); Noma (2002); Cuturi et al. (2007); Cuturi (2011). All of the mentioned kernels can be viewed from Haussler's original, visionary relation-convolution kernel framework, Haussler (1999), the existing literature provides no unifying approach to kernels for sequences: for example the relation between string kernels and dynamic time warping/global alignment kernels, or to the classical theory of time series has remained unclear. Further, the only known kernels for sequences of arbitrary objects, the dynamic time warping/global alignment kernels, are in general not positive definite.

(iii) Rough paths and stochastic processes.

Series of iterated integrals appear in diverse areas such as control theory, combinatorics, homotopy theory, physics (Feynman–Dyson–Schwinger theory) and more recently probability theory; see (Lyons, 2004, Section "Historic papers", p97). This series is treated under various names in the literature like "Magnus expansion", "time ordered" or "non-commutative exponential", or the one we follow, namely "the signature of a path". Signatures have found many applications in stochastic analysis (rough path theory, regularity structures), but their use in statistics and machine learning is very recent: Papavasiliou et al. (2011) applies it to SDE parameter estimation; Levin et al. (2013); Lyons et al. (2014) applies it to forecasting and classification; Yang et al. (2015) uses signature features as input to deep neural nets. So far, all applications are restricted to paths evolving in low-dimensional linear spaces due to the computational bottleneck given by the combinatorial explosion of signature coordinates.

1.5. Outline

Section 2 introduces the signature as a canonical feature map for path-valued data. Section 3 shows that this signature can be kernelized to get a kernel $\mathrm{k}^{\oplus}$ for paths and studies its properties. Section 4 studies the discretization to turn $\mathrm{k}^{\oplus}$ into a kernel k^{+} for sequences. Section 5 shows that classic kernels for sequences arise as special cases of our construction. Section 6 presents effective algorithms for computing k^{+} using above recursion with low-rank techniques. Section 7 presents a Numpy implementation and basic benchmarks.

1.6. Notation

Spaces, sequences and paths	
$\mathcal{X}$	an arbitrary set
$\mathcal{X}^+$	$\mathcal{X}^+ := \bigcup_{\ell \geq 1} \mathcal{X}^\ell$ sequences of arbitrary length in $\mathcal{X}$
$\mathcal{P}_\mathcal{X}$	a set of continuous paths from $[0,1]$ to $\mathcal{X}$
$\mathcal{H}$	the reproducing kernel Hilbert space, $\mathcal{H} \subset \mathbb{R}^\mathcal{X}$, of the kernel k
$\mathcal{H}^+$	$\mathcal{H}^+ := \bigcup_{\ell \geq 1} \mathcal{H}^\ell$ sequences of arbitrary length in $\mathcal{H}^+$
$\mathcal{P}_\mathcal{H}$	a set of continuous bounded variation paths from $[0,1]$ to $\mathcal{H}$
Signature features for paths and sequences in $\mathcal{H}$	
$\prod_{m\geq 0} \mathcal{H}^{\otimes m}$	the linear space of series of tensors in $\mathcal{H}$
S	the signature map $\mathrm{S} : \mathcal{P}_\mathcal{H} \to \prod_{m\geq 0} \mathcal{H}^{\otimes m}$
S^+	the discretized signature map $\mathrm{S}^+ : \mathcal{H}^+ \to \prod_{m\geq 0} \mathcal{H}^{\otimes m}$
$\langle \cdot, \cdot \rangle$	inner product on the subset of square-summable elements of $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$
$\| \cdot \|$	norm given by $\langle \cdot, \cdot \rangle$ for square summable elements of $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$
Kernels	
k	a kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
k_x with $x \in \mathcal{X}$	the element $\mathrm{k}_x := \mathrm{k}(x, \cdot) \in \mathcal{H} \subset \mathbb{R}^\mathcal{X}$
k_x with $x \in \mathcal{X}^+$	the sequence $\mathrm{k}_x := (\mathrm{k}_{x_i})_{i=1,\dots,\ell} \in \mathcal{H}^+$ with $x = (x_i)_{i=1,\dots,l}$
k_x with $x \in \mathcal{P}_\mathcal{X}$	the path $\mathrm{k}_x := (\mathrm{k}_{x(t)})_{t\in[0,1]} \in \mathcal{P}_\mathcal{H}$ with $x = (x(t))_{t\in[0,1]}$
k^+	a kernel $\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$, $\mathrm{k}^+(x,y) := \langle \mathrm{S}^+(\mathrm{k}_x), \mathrm{S}^+(\mathrm{k}_y) \rangle$
$\mathrm{k}^\oplus$	a kernel $\mathrm{k}^\oplus : \mathcal{P}_\mathcal{X} \times \mathcal{P}_\mathcal{X} \to \mathbb{R}$, $\mathrm{k}^\oplus(x,y) := \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y) \rangle$
Partitions of $[0,1]$	
π	a partition of $[0,1]$, i.e. a collection of points $0 = t_1 < \cdots < t_l = 1$
$\mathrm{mesh}(\pi)$	$\mathrm{mesh}(\pi) := \max_{i=1,\dots,l-1} \lvert t_{i+1} - t_i \rvert$
x^π for $x \in \mathcal{P}_\mathcal{X}$	$x^\pi := (x(t_i))_{i=1,\dots,l} \in \mathcal{X}^+$, the path x sampled along π

1.7. Author's contribution

Both authors contributed equally, HO is the corresponding author.

2. The signature feature map

Mapping paths to series of tensors is well-known technique in stochastic analysis. In this section we give a short introduction and highlight aspects of the signature map that make it a good feature map for paths.

2.1. Bounded variation paths

Definition 2.1 *Let $\mathcal{H}$ a Hilbert space. We denote the set of $\mathcal{H}$-valued paths of bounded variation on $[0,1]$ starting at the origin by*

$$C^1([0,1], \mathcal{H}) := \{h \in C([0,1], \mathcal{H}) \ : \ h(0) = 0, \quad \|h\|_1 < \infty\}$$

where $\|h\|_1 = \sup_\pi \sum_{i=1}^{l-1} \|h(t_{i+1}) - h(t_i)\|_{\mathcal{H}}$ *and the supremum is taken over all finite partitions* $\pi = \{(t_i)_{i=1,\dots,l} : 0 = t_1 < \cdots < t_l = 1\}$ *of* $[0,1]$.

As in the finite-dimensional case, the integral $\int_0^1 y(t)\,\mathrm{d}h(t) \in \mathcal{H}$ is defined as the limit of Riemann–Stieltjes sums, see Appendix A for details. The integral itself is in general not a scalar, but an element of the Hilbert space, $\int y\,\mathrm{d}h \in \mathcal{H}$.

Definition 2.2 *We define the sequence* $\left(\int \mathrm{d}h^{\otimes m}\right)_{m\geq 0} \in \prod_{m\geq 0} \mathcal{H}^{\otimes m}$ *as*[4]

$$\int \mathrm{d}h^{\otimes 0} := 1 \text{ and } \int \mathrm{d}h^{\otimes m} := \int_0^1 \left(\int_0^t \mathrm{d}h^{\otimes(m-1)}\right) \otimes \mathrm{d}h(t).$$

We define

$$\mathrm{S} : C^1\left([0,1], \mathcal{H}\right) \to \prod_{m\geq 0} \mathcal{H}^{\otimes m}\ ,\ h \mapsto \left(\int \mathrm{d}h^{\otimes m}\right)_{m\geq 0}$$

and call $\mathrm{S}(h)$ *the signature of the path* h *and denote* $\mathrm{S}_m(h) = \int \mathrm{d}h^{\otimes m}$.

Example 2.3 *Let* $\mathcal{H} = \mathbb{R}^2$ *and* $h(t) = (a(t), b(t))^\top$. *The first levels of the signature of* h *are*

$$\mathrm{S}_0(h) = 1, \quad \mathrm{S}_1(h) = \begin{pmatrix} \int_0^1 \mathrm{d}a(t) \\ \int_0^1 \mathrm{d}b(t) \end{pmatrix}, \quad \mathrm{S}_2(h) = \begin{pmatrix} \int_0^1 \int_0^{t_2} \mathrm{d}a(t_1)\,\mathrm{d}a(t_2) & \int_0^1 \int_0^{t_2} \mathrm{d}a(t_1)\,\mathrm{d}b(t_2) \\ \int_0^1 \int_0^{t_2} \mathrm{d}b(t_1)\,\mathrm{d}a(t_2) & \int_0^1 \int_0^{t_2} \mathrm{d}b(t_1)\,\mathrm{d}b(t_2) \end{pmatrix}.$$

The degree $m = 3$, $\mathrm{S}_3(h) \in \left(\mathbb{R}^2\right)^{\otimes 3}$, *has* 2^3*-scalar valued coordinates, etc. In general, for a path in* $\mathbb{R}^d$ *we need* d^m *scalars to describe* $\int \mathrm{d}h^{\otimes m}$ *and it is exactly this combinatorial explosion that makes the use of signatures prohibitively expensive already for moderately high-dimensional* $\mathcal{H}$ *and impossible to use for infinite-dimensional* $\mathcal{H}$ *(such as the RKHS associated with most kernels relevant in machine learning).*

Example 2.4 *For the special case of a linear path in* $\mathcal{H} = \mathbb{R}^d$, *i.e.* $h(t) = t \cdot v$ *for a fixed vector* $v \in \mathbb{R}^d$, *a direct calculation shows that*

$$\mathrm{S}_m(h) = \frac{v^{\otimes m}}{m!} = \frac{h(1)^{\otimes m}}{m!} \text{ for } m \geq 0$$

or in more concise notation[5], $\mathrm{S}(h) = \exp(v) = \exp(h(1) - h(0))$. *This explains why the signature* $\mathrm{S}(h)$ *is also known as time-ordered exponential. This analogy with the exponential function will be key motivation for the estimates in Section 4.*

4. Recall that by convention $\mathcal{H}^{\otimes 0} := \mathbb{R}$.
5. where one denotes $\boldsymbol{t} = (t_m)_m \in \prod_{m\geq 0} \mathcal{H}^{\otimes m}$ as $\boldsymbol{t} = t_0 + t_1 + t_2 + \cdots$, set $\exp(\boldsymbol{t}) = \sum_{m\geq 0} \frac{\boldsymbol{t}^{\otimes m}}{m!}$ and identify $v \in \mathbb{R}^d$ as $\boldsymbol{t} = 0 + v + 0 + 0 + \cdots$.

2.2. The signature as a canonical feature map

The signature $\mathrm{S}(h)$ describes a path h by an element in the linear space $\prod_{m\geq 0}\mathcal{H}^{\otimes m}$ and the following properties make it a good feature map

Theorem 1

1. *The map $h \mapsto \mathrm{S}(h)$ is continuous,*
2. *The map $h \mapsto \mathrm{S}(h)$ is injective up to tree-like equivalence*[6]*,*
3. *For any $f \in C(K, \mathbb{R})$, $K \subset C^1([0,1], \mathcal{H})$ compact, and $\epsilon > 0$ there exists a $\ell \in \bigoplus_{m\geq 0}\mathcal{H}^{\otimes m}$ such that*

$$\sup_{h\in K} |f(h) - \langle \ell, \mathrm{S}(h)\rangle| < \epsilon.$$

The proof is classic in stochastic analysis and given in Appendix A; the key insight is that the space of linear combinations of iterated integrals forms an algebra. To sum up

1. the signature features are a *mathematically faithful representation of the underlying path-valued data h.*
2. *linear combinations* of signature coordinates *approximate continuous functions of paths* arbitrarily well. In the kernel learning context this is known as "universality".
3. the signature features are *sequentially ordered analogues to moments.* Similar to polynomials, the feature space has the natural grading by m designating the "polynomial degree".

3. Signature kernels

We come back to our original motivation: we are given a RKHS $(\mathcal{H}, \mathrm{k})$ on $\mathcal{X}$ and we want to transform this into a RKHS for paths in $\mathcal{X}$. Therefore recall that the reproducing property implies that

$$\mathrm{k} : \mathcal{X}\times\mathcal{X} \to \mathbb{R} \text{ equals } \mathrm{k}(u,v) = \langle \mathrm{k}_u, \mathrm{k}_v\rangle_{\mathcal{H}}$$

and that one may think of $\mathcal{X} \ni u \mapsto \mathrm{k}_u(\cdot) := \mathrm{k}(u,\cdot) \in \mathcal{H}$ as the (canonical) feature map induced by the kernel k. Now given a path $x \in \mathcal{P}_{\mathcal{X}}$ we lift it to a path[7] $\mathrm{k}_x \in \mathcal{P}_{\mathcal{H}}$ given as

6. We call h, h' tree-like equivalent if the concatenation $h \sqcup h'^{-1}$ is tree-like. A path $z \in C^1([a,b],\mathcal{H})$ is called tree-like if there exists a continuous map $f : [a,b] \to [0,\infty)$ with $f(0) = f(T) = 0$ and $|z(t) - z(s)| \leq f(s) + f(t) - 2\inf_{u\in[s,t]} f(u)$ for all $s \leq t \in [a,b]$. The key remark is that being tree-like equivalent is a very strong assumption, typically not seen in real data. Even if presented with a tree-like path, adding time as extra coordinate transforms the path to a non-treelike path (that is, working with $t \mapsto (t, h(t))$ instead of $t \mapsto h(t)$).
7. Recall the slight clash of notation $\mathrm{k}_u \in \mathcal{H}$ for $u \in \mathcal{X}$ and $\mathrm{k}_x \in \mathcal{P}_{\mathcal{H}}$ if $x \in \mathcal{P}_{\mathcal{X}}$, but the meaning will always be clear from the context. In fact, later we are also going to use $\mathrm{k}_x \in \mathcal{H}^+$ if $x \in \mathcal{X}^+$.

$t \mapsto \mathrm{k}_x(t) := \mathrm{k}_{x(t)}(\cdot) := \mathrm{k}(x(t), \cdot)$. The results of Section 2 suggest $\mathrm{S}(\mathrm{k}_x)$ as features for k_x, that is the feature map

$$\mathcal{P}_X \ni x \mapsto \mathrm{S}(\mathrm{k}_x) \in \prod_{m \geq 0} \mathcal{H}^{\otimes m}. \tag{2}$$

Theorem 1 guarantees that $\mathrm{S}(\mathrm{k}_x)$ captures all the information about $x \in \mathcal{P}_X$ that can be obtained by looking at the path x through the kernel k.

Most RKHS $\mathcal{H}$ in machine learning are infinite dimensional which makes this approach infeasible; even for finite dimensional $\mathcal{H}$ we suffer from the combinatorial explosion of the number of signature coordinates, Example 2.3. The main result of this section is that the feature map (2) can be complety kernelized, that is by defining

$$(x, y) \mapsto \mathrm{k}^{\oplus}(x, y) = \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y) \rangle$$

we get a kernel for paths in X that can be very efficiently evaluated using only $\mathrm{k}\left(x(s), y(t)\right)$ for $s, t \in [0, 1]$. That is, only evaluations of the "static" kernel $\mathrm{k} : X \times X \to \mathbb{R}$.

We henceforth restrict attention to positive definite kernels on X and a set of paths $\mathcal{P}_X$ that lift to bounded variation paths. All our arguments extend to unbounded variation such as semi-martingales, diffusion processes, Markov processes or Gaussian processes and we give the needed modifications in Appendix B.

Assumption 3.1 *Throughout the remainder of this article we assume that* $\mathrm{k} : X \times X \to \mathbb{R}$ *is continuous and positive definite. We denote with* $\mathcal{H}$ *the associated RKHS. Further, we denote with* $\mathcal{P}_X \subset X^{[0,1]}$ *a set of paths in a topological space* X *and* $(\mathcal{H}, \mathrm{k})$ *a RKHS on* X *such that*

$$\{t \mapsto \mathrm{k}_{x(t)} : x \in \mathcal{P}_X\} \subset C^1([0, 1], \mathcal{H}).$$

This yields the following kernel,

Definition 3.2 *We call*

$$\mathrm{k}^{\oplus} : \mathcal{P}_X \times \mathcal{P}_X \to \mathbb{R}\,, \quad (x, y) \mapsto \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y) \rangle,$$

the signature kernel *over* k.

Theorem 2 *The signature kernel over* k

$$\mathrm{k}^{\oplus} : \mathcal{P}_X \times \mathcal{P}_X \to \mathbb{R}, \quad \mathrm{k}^{\oplus}(x, y) = \langle \mathrm{S}\left(\mathrm{k}_x\right), \mathrm{S}\left(\mathrm{k}_x\right) \rangle$$

1. *is a positive definite kernel,*
2. $\mathrm{k}^{\oplus}(x, y) = \sum_{m \geq 0} \int_{\substack{s_1 < \cdots < s_m \\ t_1 < \cdots < t_m}} \prod_{i=1}^{m} \mathrm{d}\kappa(s_i, t_i)$ *with* κ *defined as*

$$\kappa\left([s, t] \times [u, v]\right) := \mathrm{k}\left(x(t), y(v)\right) - \mathrm{k}\left(x(s), y(v)\right) - \mathrm{k}\left(x(t), y(u)\right) + \mathrm{k}\left(x(s), y(u)\right)$$

and κ *is a signed Borel measure on* $[0, 1]^2$ *for every* $x, y \in \mathcal{P}_X$.

Proof Point (1) follows since $\mathrm{k}^{\oplus}$ is given as an inner product, see Schölkopf and Smola (2002). Using the definition of $\mathrm{k}^{\oplus}$ we calculate

$$\begin{aligned}
\mathrm{k}^{\oplus}(x,y) &= \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y)\rangle = \sum_{m\geq 0} \langle \int \mathrm{dk}_x^{\otimes m}, \int \mathrm{dk}_y^{\otimes m}\rangle_{\mathcal{H}^{\otimes m}} \\
&= \sum_{m\geq 0} \langle \int_0^1 \left(\int_0^{s_{m-1}} \mathrm{dk}_x^{\otimes (m-1)} \right) \otimes \mathrm{dk}_{x(s_m)}, \int_0^1 \left(\int_0^{t_{m-1}} \mathrm{dk}_y^{\otimes (m-1)} \right) \otimes \mathrm{dk}_{y(t_m)} \rangle \\
&= 1 + \sum_{m\geq 1} \int_{\substack{s_m\in[0,1]\\ t_m\in[0,1]}} \langle \int_0^{s_m} \mathrm{dk}_x^{\otimes (m)}, \int_0^{t_{m-1}} \mathrm{dk}_y^{\otimes (m-1)} \rangle \langle \mathrm{dk}_{x(s_m)}, \mathrm{dk}_{y(t_m)}\rangle \\
&= \vdots \\
&= 1 + \sum_{m\geq 1} \int_{\substack{s_m\in[0,1]\\ t_m\in[0,1]}} \int_{\substack{s_{m-1}\in[0,s_m]\\ t_{m-1}\in[0,t_m]}} \cdots \int_{\substack{s_1\in[0,s_2]\\ t_1\in[0,t_2]}} \langle \mathrm{dk}_{x(s_1)}, \mathrm{dk}_{y(t_1)}\rangle \cdots \langle \mathrm{dk}_{x(s_m)}, \mathrm{dk}_{y(t_m)}\rangle.
\end{aligned} \tag{3}$$

Now consider two paths x, y that are piecewise linear, that is there are time points $s_1 \leq \cdots \leq s_k$ such that $\dot{x}$ is constant on intervals (s_i, s_{i+1}) and analogous for y there exists $t_1 \leq \cdots \leq t_l$ such that $\dot{y}$ is constant on all intervals (t_i, t_{i+1}). Then

$$\begin{aligned}
&\int_{(s_1,s_k)\times(t_1,t_l)} \langle \mathrm{dk}_{x(s)}, \mathrm{dk}_{y(t)}\rangle_{\mathcal{H}} \\
&= \sum_{\substack{1\leq i\leq k-1\\ 1\leq j\leq l-1}} \int_{(s_i,s_{i+1})\times(t_j,t_{j+1})} \langle \mathrm{dk}_{x(s)}, \mathrm{dk}_{y(t)}\rangle_{\mathcal{H}} \\
&= \sum_{\substack{1\leq i\leq k-1\\ 1\leq j\leq l-1}} \langle \mathrm{k}_{x(s_{i+1})} - \mathrm{k}_{x(s_i)}, \mathrm{k}_{y(t_{j+1})} - \mathrm{k}_{y(t_j)}\rangle_{\mathcal{H}} \\
&= \sum_{\substack{1\leq i\leq k-1\\ 1\leq j\leq l-1}} \mathrm{k}(x(s_{i+1}), y(t_{j+1})) - \mathrm{k}(x(s_i), y(t_{j+1})) - \mathrm{k}(x(s_{i+1}), y(t_j)) + \mathrm{k}(x(s_i), y(t_j))
\end{aligned}$$

where we used the reproducing property $\langle \mathrm{k}_{x(s)}, \mathrm{k}_{y(t)}\rangle = \mathrm{k}(x(s), y(t))$ for the last equality. Note that

$$\kappa([r,s]\times[u,v]) := \mathrm{k}(x(s_{i+1}), y(t_{j+1})) - \mathrm{k}(x(s_i), y(t_{j+1})) - \mathrm{k}(x(s_{i+1}), y(t_j)) + \mathrm{k}(x(s_i), y(t_j))$$

defines a signed measure on $[0,1]^2$, hence above reads

$$\int_{(s_1,s_k)\times(t_1,t_l)} \langle \mathrm{dk}_{x(s)}, \mathrm{dk}_{y(t)}\rangle_{\mathcal{H}} = \sum_{\substack{1\leq i\leq k\\ 1\leq j\leq l}} \kappa([s_1,s_k]\times[t_1,t_l]) = \int_{(s_1,s_k)\times(t_1,t_l)} \mathrm{d}\kappa_{x,y}(s,t).$$

Combined with the identity (3) this shows (2) for piecewise constant paths. For general x, y fix a sequence $\{(s_i^n) : 0 = s_1 < \cdots < s_{l+1}^n = 1\} \subset [0,1]^n$ such that $\max_i |s_{i+1}^n - s_i^n| \to 0$

as $n \to \infty$ and denote with x^n, resp. y^n, the piecewise linear path given by interpolating between points $(x(s_i^n))_i$, resp. $(y(s_j^n))_j$. Then

$$\kappa_{x^n,y^n}([r,s]\times[u,v]) \to \kappa_{x,y}([r,s]\times[u,v])$$

for all $r < s, u < v$ directly by the definition of κ. Thus κ_{x^n,y^n} converges weakly to $\kappa_{x,y}$ On the other hand, $\langle \mathrm{S}(x^n), \mathrm{S}(y^n)\rangle$ converges to $\langle \mathrm{S}(x), \mathrm{S}(y)\rangle$ which finishes the proof by sending $n \to \infty$ ∎

The equality in Point (2) in Theorem 2 suggests the expansion

$$\mathrm{k}^{\oplus}(x,y) = 1 + \int_{(s_1,t_1)\in(0,1)\times(0,1)} \left(1 + \int_{(s_2,t_2)\in(0,s_1)\times(0,t_1)} (1+\cdots)\,\mathrm{d}\kappa_{x,y}(s_2,t_2)\right) \mathrm{d}\kappa_{x,y}(s_1,t_1). \tag{4}$$

This recursive structure will be the key for an efficient computation.

4. Discretization and recursion: from paths to sequences

To make the kernel $\mathrm{k}^{\oplus} : \mathcal{P}_{\mathcal{X}} \times \mathcal{P}_{\mathcal{X}} \to \mathbb{R}$ useful for real-world applications, we need to address the following two points:

1. (Sequences). We do not observe the full path $(x(t))_{t\in[0,1]} \in \mathcal{P}_{\mathcal{X}}$ but only a sequence $x^\pi = (x(t_i))_{i=1}^l \in \mathcal{X}^+$ for a partition $\pi = \{0 = t_1 < \cdots < t_\ell = 1\}$. The partition π might even vary from path to path.

2. (Computational cost). Computing the kernel $\mathrm{k}^{\oplus} : \mathcal{P}_{\mathcal{X}} \times \mathcal{P}_{\mathcal{X}} \to \mathbb{R}$ by evaluating it as an inner product is computationally prohibitive even with $\mathcal{X} = \mathbb{R}^d = \mathcal{H}$ and the linear kernel $\mathrm{k}(\cdot,\cdot) = \langle\cdot,\cdot\rangle_{\mathbb{R}^d}$ is used if d is large (Example 2.3). It is infeasible for standard kernels such as the Gaussian kernel that have an infinite-dimensional RKHS.

To address point (1), we approximate the signature map

$$x \mapsto \mathrm{S}(\mathrm{k}_x) = \left(\int \mathrm{dk}_x^{\otimes m}\right)_{m\geq 0}$$

by discretization of the integrals. This gives a feature map S^+ for sequences

$$x^\pi \mapsto \mathrm{S}^+(\mathrm{k}_{x^\pi}) = \left(\sum_{1\leq i_1<\cdots<i_m\leq \ell-1} \nabla_{i_1}\mathrm{k}_x \otimes\cdots\otimes \nabla_{i_m}\mathrm{k}_x\right)_{m\geq 0}$$

where we use the notation $\nabla_i \mathrm{k}_x := \mathrm{k}_{x(t_{i+1})} - \mathrm{k}_{x(t_i)}$ and $\mathrm{k}_{x^\pi} = (\mathrm{k}_{x(t_i)})_{i=1}^l \in \mathcal{H}^+$. A simple calculation shows that this is equivalent to

$$x^\pi \mapsto \mathrm{S}^+(\mathrm{k}_{x^\pi}) = \prod_{i=1}^{\ell}(1 + \nabla_i \mathrm{k}_{x_i^\pi})$$

if $\prod$ denotes the linear extension[8] of the tensor product to the linear space $\prod_{m\geq 0}\mathcal{H}^{\otimes m}$ and this is the definition we are going to use. The kernel k^+ is defined as the inner product of this feature map

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad (x, y) \mapsto \langle \mathrm{S}^+(\mathrm{k}_x), \mathrm{S}^+(\mathrm{k}_y)\rangle$$

To address Point (2), we make heavy use of the recursive expansion (4) for $\mathrm{k}^{\oplus}$ and truncate after a given degree m (that is usually determined by cross-validation to avoid overfitting in analogy to the classic polynomial kernel).

4.1. Discretized signatures

Definition 4.1 *We call the map*

$$\mathrm{S}^+ : \mathcal{H}^+ \to \prod_{m\geq 0}\mathcal{H}^{\otimes m}, \quad (h_i)_{i=1}^l \mapsto \prod_{i=1}^{l}(1+\nabla_i h)$$

the discretized signature map (of order 1*).*

To prove quantitative results about the quality of this discretization, we need to understand how fast $\mathrm{S}^+(\mathrm{k}_{x^\pi})$ approximates $\mathrm{S}(\mathrm{k}_x)$ as $\mathrm{mesh}(\pi) := \max_i |t_{i+1} - t_i| \to 0$. We first study this approximation of paths by sequences for general paths $h \in \mathcal{P}_{\mathcal{H}}$.

Definition 4.2 *Define for* $d \geq 1$

$$\exp_d : \mathbb{R}^d \to \mathbb{R}, \quad v = (v_i)_{i=1}^d \mapsto \prod_{i=1}^{d}(1+v_i)$$

Define for a partition $\pi = (t_i)_{i=1}^{\ell}$ *with* $0 = t_1 < \cdots < t_\ell = 1$

$$\|h\|_{(\pi)} : C^1([0,1],\mathcal{H}) \to \mathbb{R}, \quad h \mapsto \left(\|h_{[t_1,t_2]}\|_1, \ldots, \|h_{[t_{\ell-1},t_\ell]}\|_1\right) \in \mathbb{R}^{\ell-1}$$

where $h_{[a,b]}$ *denotes the restriction of* $h \in C^1([0,1],\mathcal{H})$ *to* $C^1([a,b],\mathcal{H})$*.*

Using this notation, the main result of this section reads

Theorem 3 *For* $h \in C^1([0,1],\mathcal{H})$ *and a partition* $\pi = (t_i)_{i=1}^{\ell}$*,* $0 = t_1 < \cdots < t_\ell = 1$*,*

$$\left\|\mathrm{S}^+(h^\pi) - \mathrm{S}(h)\right\| \leq \exp\left(\|h\|_1\right) - \exp_1\left(\|h\|_{(\pi)}\right),$$

where $h^\pi = (h(t_i))_{i=1,\ldots,\ell} \in \mathcal{H}^+$*.*

As a consequence, we get explicit convergence rates.

8. For $\boldsymbol{s}, \boldsymbol{t} \in \prod_{m\geq 0}\mathcal{H}^{\otimes m}$ with $s_0 = t_0 = 1$ define $\boldsymbol{s}\cdot\boldsymbol{t} \in \prod_{m\geq 0}\mathcal{H}^{\otimes m}$ as having the m-element equal to $\sum_{i=0}^{m} s_i t_{m-i}$. If we denote $\boldsymbol{s} = (s_0, s_1, \ldots)$ as $\boldsymbol{s} = s_0 + s_1 + \cdots$, then this reads $\boldsymbol{s}\cdot\boldsymbol{t} = 1 + s_1 + t_1 + s_2 + s_1 t_1 + t_2 + \cdots$ and $\prod(1+\nabla_i h)$ refers to this product. Informally speaking, we multiply series of tensors in the same way we multiply polynomials.

Corollary 4.3 *With the same notation as in Theorem 3 above, it holds that*

$$\left\|\mathrm{S}^+(h^\pi) - \mathrm{S}(h)\right\| \leq \|h\|_1 e^{\|h\|_1} \cdot \max_{i=1,\dots,\ell-1} \|h_{[t_i,t_{i+1}]}\|_1.$$

Hence, the convergence $\lim_{\|h\|_{(\pi)} \to \|h\|_1} \mathrm{S}^+(h^\pi) = \mathrm{S}(h)$*, is of order* $O\left(\max_i \|h_{[t_i,t_{i+1}]}\|_1\right)$.

Corollary 4.4 *With the same notation as in Theorem 3 above, it holds that if* π *is chosen such that* $\|h_{[t_i,t_{i+1}]}\|_1 = \frac{\|h\|_1}{\ell-1}$ *for all* $i = 1, \dots, \ell-1$*, then one has the asymptotically tight bound*

$$\left\|\mathrm{S}^+(h^\pi) - \mathrm{S}(h)\right\| \leq \frac{e^{\|h\|_1}}{\ell - 1}\left(1 + \frac{\|h\|_1^{\ell-1}}{(\ell-3)!}\right).$$

The proofs of Theorem 3 and Corollaries 4.3 and 4.4 are given in Appendix A.

Remark 4.5 (About geometric approximations) *The reader familiar with path signatures will note that the approximations* $\mathrm{S}^+(h^\pi)$ *to* $\mathrm{S}(h)$ *are not group-like elements of the tensor (Hopf) algebra* $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$*, thus "non-geometric" in the sense that there can not exist a path* $\hat{h}$ *such that* $\mathrm{S}^+(h^\pi) = \mathrm{S}(\hat{h})$*. However, as we will see in the following sections, from a computational perspective non-geometric approximations have many benefits. In fact, the "non-geometric" approximations of this section recover classic machine learning constructions, see Section 5. Nevertheless, one can modify the discussion of this section to produce geometric approximations and we carry this out in Appendix B; see also Remark B.6.*

4.2. Discretized signature kernels

Definition 4.6 *Given* $x = (x_i)_{i=1}^l \in \mathcal{X}^+$ *denote* $\mathrm{k}_x = (\mathrm{k}_{x_i})_{i=1}^l \in \mathcal{H}^+$*. We refer to*

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad \mathrm{k}^+(x,y) = \langle \mathrm{S}^+(\mathrm{k}_x), \mathrm{S}^+(\mathrm{k}_y)\rangle$$

as the discretized signature kernel (or sequentialization) over k *of order* 1*.*

Theorem 4 *The discretized signature kernel over* k*,*

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad \mathrm{k}^+(x,y) = \langle \mathrm{S}^+\left(\mathrm{k}_x\right), \mathrm{S}^+\left(\mathrm{k}_x\right)\rangle$$

1. *is a positive definite kernel,*
2. $\mathrm{k}^+(x,y) = \sum_{m\geq 0} \sum_{\substack{1\leq i_1 < \dots < i_m < |x| \\ 1 \leq j_1 < \dots < j_m < |y|}} \prod_{r=1}^m \nabla_{i_r,j_r} \mathrm{k}(x,y)$*,*
3. $\mathrm{k}^+(x,y) = 1 + \sum_{\substack{i_1 \geq 1 \\ j_1 \geq 1}} \nabla_{i_1,j_1} \mathrm{k}(x,y) \cdot \left(1 + \sum_{\substack{i_2 > i_1 \\ j_2 > j_1}} \nabla_{i_2,j_2} \mathrm{k}(x,y) \cdot \left(1 + \sum_{\substack{i_3 > j_2 \\ j_3 > j_2}} \dots\right)\right)$*,*

where we use the notation $x = (x_i)_{i=1}^{|x|}, y = (y_i)_{i=1}^{|y|} \in \mathcal{X}^+$ *and*

$$\nabla_{i,j} \mathrm{k}(x,y) := \mathrm{k}(x_{i+1}, y_{j+1}) + \mathrm{k}(x_i, y_j) - \mathrm{k}(x_i, y_{j+1}) - \mathrm{k}(x_{i+1}, y_j).$$

Proof The first point follows, since k^+ is an inner product. For the second point note that $\nabla_i \mathrm{k}_x \equiv \mathrm{k}_{x_{i+1}} - \mathrm{k}_{x_i}$, $\nabla_i \mathrm{k}_y \equiv \mathrm{k}_{y_{j+1}} - \mathrm{k}_{y_j}$ and that by the reproducing property $\langle \nabla_i \mathrm{k}_x, \nabla_j \mathrm{k}_y \rangle = \nabla_{i,j} \mathrm{k}(x, y)$. The identity then follows from multiplying out

$$\langle \mathrm{S}^+(\mathrm{k}_x), \mathrm{S}^+(\mathrm{k}_y) \rangle = \langle \prod_i (1 + \nabla_i \mathrm{k}_x), \prod_j (1 + \nabla_j \mathrm{k}_y) \rangle$$

and using the definition of the inner product on $\mathcal{H}^{\otimes m}$. The last point follows by rearranging the summation. ∎

We can now combine this with Theorem 3 to quantify how fast the kernel k^+ for sequences approximates the kernel $\mathrm{k}^\oplus$ for paths as the mesh of partitions gets finer.

Corollary 4.7 *Let $x, y \in \mathcal{P}_\mathcal{X}$, and $\pi = (s_i)_{i=1}^k$ with $0 = s_1 < \cdots < s_k = 1$ and $\rho = (t_j)_{j=1}^l$ with $0 = t_1 < \cdots < t_l = 1$. Then,*

$$\left| \mathrm{k}^+(x^\pi, y^\rho) - \mathrm{k}^\oplus(x, y) \right| \leq 4e^{\|x\|_1 + \|y\|_1} - 2e^{\|x\|_{(\pi)} + \|y\|_1} - 2e^{\|x\|_1 + \|y\|_{(\rho)}}.$$

In particular,

$$\lim_{\substack{\|x\|_{(\pi)} \to \|x\|_1 \\ \|y\|_{(\rho)} \to \|y\|_1}} \mathrm{k}^+(x^\pi, y^\rho) = \mathrm{k}^\oplus(x, y)$$

where convergence is uniform of order $O\left(\max_i \|x_{[s_i, s_{i+1}]}\|_1 + \max_j \|y_{[t_j, t_{j+1}}\|_1\right)$.

Proof It holds that

$$\begin{aligned} \mathrm{k}^+(x^\pi, y^\rho) - \mathrm{k}^\oplus(x, y) =& \langle \mathrm{S}^+(\mathrm{k}_x^\pi), \mathrm{S}^+(\mathrm{k}_y^\rho) \rangle - \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y) \rangle \\ =& \langle \mathrm{S}^+(\mathrm{k}_x^\pi), \mathrm{S}^+(\mathrm{k}_y^\rho)) - \mathrm{S}(\mathrm{k}_y) \rangle + \langle \mathrm{S}^+(\mathrm{k}_x^\pi) - \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y) \rangle. \end{aligned}$$

The Cauchy–Schwarz inequality implies that

$$|\langle \mathrm{S}^+(\mathrm{k}_x^\pi), \mathrm{S}^+(\mathrm{k}_y^\rho)) - \mathrm{S}(\mathrm{k}_y) \rangle| \leq \| \mathrm{S}^+(\mathrm{k}_x^\pi) \| \cdot \| \mathrm{S}^+(\mathrm{k}_y^\rho) - \mathrm{S}(\mathrm{k}_y) \|.$$

Theorem 3 implies

$$\| \mathrm{S}^+(\mathrm{k}_x^\pi) \| \cdot \| \mathrm{S}^+(\mathrm{k}_y^\rho)) - \mathrm{S}(\mathrm{k}_y) \| \leq 2 \exp(\|x\|_1) \cdot \left(\exp(\|y\|_1 - \exp(\|y\|_\rho) \right)$$

and similarly, one obtains

$$\| \langle \mathrm{S}^+(\mathrm{k}_y^\rho), \mathrm{S}^+(\mathrm{k}_x^\pi) - \mathrm{S}(\mathrm{k}_x) \rangle \| \leq 2 \exp(\|y\|_1) \cdot (\exp(\|x\|_1) - \exp(\|x\|_\pi)).$$

Putting all (in-)equalities together yields the main claim, the convergence statement follows from Theorem 3 (ii). ∎

4.3. Truncation and variations on a theme

The final step to get a practically useful kernel for sequences is to truncate the discrete signature features $\mathrm{S}^+(\mathrm{k}_x) \in \prod_{m\geq 0} \mathcal{H}^{\otimes m}$ at a given degree $m \geq 1$. In analogy with the classic polynomial features (resp. kernel), this not only serves a computational purpose but is also necessary to avoid overfitting; in applications the truncation degree m will be typically determined by cross-validation.

Definition 4.8 *Given an integer $m \geq 1$, we call*

$$\mathrm{k}^+_m : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad \mathrm{k}^+_m(x,y) = \langle \mathrm{S}^+(\mathrm{k}_x), \mathrm{S}^+(\mathrm{k}_y)\rangle_m$$

the discretized signature kernel (or sequentialization) over k *truncated at degree m. Here we use the notation $\mathrm{k}_x = (\mathrm{k}_{x_i})_i \in \mathcal{H}^+$ for $x = (x_i)_i \in \mathcal{X}^+$ and $\langle\cdot,\cdot\rangle_m$ for the inner product truncated*[9] *at m.*

Corollary 4.9 *The discretized signature kernel k^+_m over* k *truncated at degree m,*

$$\mathrm{k}^+_m : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad \mathrm{k}^+_m(x,y) = \langle \mathrm{S}^+(\mathrm{k}_x), \mathrm{S}^+(\mathrm{k}_y)\rangle_m,$$

1. *is a positive definite kernel,*
2. $\mathrm{k}^+_m(x,y) = \sum_{d=1}^m \sum_{\substack{1\leq i_1<\cdots<i_d<|x| \\ 1\leq j_1<\cdots<j_d<|y|}} \prod_{r=1}^d \nabla_{i_r,j_r} \mathrm{k}(x,y)$
3. $\mathrm{k}^+_m(x,y) = 1 + \sum_{\substack{|x|>i_1\geq 1 \\ |y|>j_1\geq 1}} \nabla_{i_1,j_1} \mathrm{k}(x,y) \cdot \left(1 + \cdots (1 + \sum_{\substack{|x|>i_m>j_{m-1} \\ |y|>j_m>j_{m-1}}} \nabla_{i_m,j_m} \mathrm{k}(x,y))\right)$

where we denote $x = (x_i)_{i=1}^{|x|}, y = (y_i)_{i=1}^{|y|} \in \mathcal{X}^+$ and

$$\nabla_{i,j} \mathrm{k}(x,y) := \mathrm{k}(x_{i+1}, y_{j+1}) + \mathrm{k}(x_i, y_j) - \mathrm{k}(x_i, y_{j+1}) - \mathrm{k}(x_{i+1}, y_j).$$

While above follows as simple corollary from the discussion so far, the identity Point (3) will be the key to an efficient algorithm; it is clearly more efficient than computing first $\mathrm{S}^+(x)$ and $\mathrm{S}^+(y)$ and then their inner product; but it is also much more efficient than the identity in Point (2) since it only uses $\nabla_{i,j} \mathrm{k}(x,y)$ for each i,j once.

Remark 4.10 (Variations on a theme) *Modifications of the above are possible. For example,*

(i) One can omit the first differences, that is consider the sequentialization

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (x,y) \mapsto 1 + \sum_{\substack{m\geq 1 \\ 1\leq i_1<\cdots<i_m\leq|x| \\ 1\leq j_1<\cdots<j_m\leq|y|}} \prod_{k=1}^m \mathrm{k}(x_{i_k}, y_{j_k}).$$

In feature space, this amounts to replacing the path $(\mathrm{k}_t)_{t\in[0,1]}$ by the path $(\int_0^t \mathrm{k}_s \,\mathrm{d}s)_{t\in[0,1]}$ and calculating the signature of the latter.

9. $\langle (s_0, s_1, \ldots), (t_0, t_1, \ldots)\rangle_m := \langle (s_0, s_1, \ldots, s_m, 0, \ldots), (t_0, t_1, \ldots, t_m, 0, \ldots)\rangle$

(ii) More generally, one may replace for each m, the inner product $\prod_{k=1}^{m} \mathrm{k}(x_{i_k}, y_{j_k})$ by a kernel $\mathrm{k}^m : \mathcal{X}^m \times \mathcal{X}^m \to \mathbb{R}$. This leads to a sequentialization of the family of kernels $(\mathrm{k}^m)_m$ given as

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (x, y) \mapsto \sum_{\substack{m \geq 1 \\ 1 \leq i_1 < \cdots < i_m \leq |x| \\ 1 \leq j_1 < \cdots < j_m \leq |y|}} \mathrm{k}^m((x_{i_1}, \ldots, x_{i_m}), (y_{j_1}, \ldots, y_{j_m})).$$

This corresponds to choosing different kernels on different levels of the tensor algebra, e.g., re-normalization.

(iii) One may (additionally) opt to remember the position of elements in the sequence. This can be done by mapping sequences in $\mathcal{X}^+$ to a position-remembering sequence in $(\mathcal{X} \times \mathbb{N})^+$ first, and then carrying out the sequentialization in (ii) for the family of kernels $(\mathrm{k}^m)_m$ where $\mathrm{k}^m : (\mathcal{X} \times \mathbb{N})^+ \times (\mathcal{X} \times \mathbb{N})^+ \to \mathbb{R}$ is defined as

$$\kappa\left((i_1, \ldots, i_m), (j_1, \ldots, j_m)\right) \cdot \mathrm{k}^m\left((x_{i_1}, \ldots, x_{i_m}), (y_{j_1}, \ldots, y_{j_m})\right).$$

and $\kappa : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{R}$ denotes an arbitrary positive definite kernel. Further, one can replace the strict inequalities $<$ in the summation over $1 \leq i_1 < \cdots < i_m \leq |x|$ and $1 \leq j_1 < \cdots < j_m \leq |y|$ by $\leq$.

Further, Assumption 3.1, especially the continuity of k *can be dropped and any of above sequentializations gives a kernel for sequences (but convergence to a kernel on paths is no longer guaranteed). There are many other modifications and above choices, while minor, seem somewhat arbitrary. The above have their justification in explaining prior art, see Section 5 below.*

5. Revisiting classic kernels for sequences: string, alignment and convolution kernels

In this section we show that the discretized signature kernel is closely related to the existing kernels for sequential data, in the sense that classic kernels for sequences arise as special cases of our approach, namely:

(a) String kernels Lohdi et al. (2002); Leslie and Kuang (2004) can be seen as a case of discretized signature kernels.

(b) The global alignment kernel Cuturi et al. (2007); Cuturi (2011) can be obtained from a special case of the discretized signature kernel, by deleting some smaller terms, in the process destroying positive definiteness.

(c) The discretized signature kernel can be related to the framework of the relation-convolution kernel Haussler (1999) with the relation "being a subsequence".

The results of this section are not important for using our method as presented in the previous and following section, but it might be of interest to researchers who are familiar with above kernels.

5.1. The string kernel

The string kernel is a kernel for sequences in a finite set $\mathcal{X}$; in this context, the elements of $\mathcal{X}$ are usually called letters and $\mathcal{X}$ is called the alphabet. Two important examples are the Latin alphabet $\mathcal{X} = \{A, B, \dots, Z, a, b, \dots, z\}$ for classic text mining, or $\mathcal{X}$ being DNA nucleotides, proteins, etc., in bioinformatics.

Definition 5.1 (Definition 1, page 423 of Lohdi et al. (2002)) *Fix a finite set $\mathcal{X}$ and a parameter $\lambda \in [0,1]$. The string kernel on $\mathcal{X}$ with parameter λ is defined as*

$$\mathrm{k}_{\mathrm{string}} : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R} \quad (x,y) \mapsto \sum_{w \in \mathcal{X}^+} \Phi_w(x)\Phi_w(y)$$

where

$$\Phi_w(x) = \sum_{1 \le i_1 < \cdots < i_m \le r} \lambda^{|i_m - i_1 + 1|} 1_{(x_{i_1}, \dots, x_{i_k}) = w} \text{ for } w \in \mathcal{X}^+.$$

Proposition 5.2 *Fix a finite set $\mathcal{X}$ and a parameter $\lambda \in [0,1]$. Define a kernel*

$$\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad \mathrm{k}(a,b) = 1_{a=b}.$$

Then the sequentialization of k *given in Remark 4.10 (i), equals the string kernel with parameter $\lambda = 1$ up to an additive constant 1, that is $\mathrm{k}^+(x,y) = 1 + \mathrm{k}_{\mathrm{string}}(x,y)$ for all $x, y \in \mathcal{X}^+$.*

Proof We apply Theorem 4 in the variation given in Remark 4.10 (i) to the kernel k to get

$$\begin{aligned}
\mathrm{k}^+(x,y) &= 1 + \sum_{m \ge 1} \sum_{\substack{1 \le i_1 < \cdots < i_m \le |x| \\ 1 \le j_1 < \cdots < j_m \le |y|}} \prod_{r=1}^{m} \mathrm{k}(x_{i_r}, y_{j_r}) = 1 + \sum_{m \ge 1} \sum_{\substack{1 \le i_1 < \cdots < i_m \le |x| \\ 1 \le j_1 < \cdots < j_m \le |y|}} 1_{x_{i_1} = y_{j_1}} \cdots 1_{x_{i_m} = y_{j_m}} \\
&= 1 + \sum_{m \ge 1} \sum_{\substack{1 \le i_1 < \cdots < i_m \le |x| \\ 1 \le j_1 < \cdots < j_m \le |y|}} 1_{(x_{i_1}, \dots, x_{i_m}) = (y_{j_1}, \dots, y_{j_m})} \\
&= 1 + \sum_{m \ge 1} \sum_{w \in \mathcal{X}^+, |w| = m} \Phi_w(x)\Phi_w(y) = 1 + \sum_{w \in \mathcal{X}^+} \Phi_w(x)\Phi_w(y) = 1 + \mathrm{k}_{\mathrm{string}}(x,y)
\end{aligned}$$

■

More generally, for any $\lambda \in [0,1]$ one can apply the κ-sequentialization given in Remark 4.10 (iii), to the kernel $\mathrm{k}(a,b) = 1_{a=b}$ and

$$\kappa((i_1, \dots, i_m), (j_1, \dots, j_m)) = \lambda^{i_m - i_1 + 1} \lambda^{j_m - j_1 + 1}$$

to recover the string kernel $\mathrm{k}_{\mathrm{string}}$ for the parameter λ; this follows as above. Similarly, various variants of the vanilla string kernel such as gappy kernels, Leslie and Kuang (2004) can be recovered[10]. Note that the $+1$ in $\mathrm{k}_{\mathrm{string}} + 1 = \mathrm{k}^+$ arises since in string kernels one usually does not count the empty word (if we count the empty word then $\mathrm{k}^+ = \mathrm{k}_{\mathrm{string}}$). However, this is merely convention and of non-importance for the Gram matrix.

10. As one of the reviewers pointed out, a very general framework for sequences that encompasses string kernels are so-called rational kernels.

Remark 5.3 *Implicit in above discussion is the identification of a string formed from an alphabet $\mathcal{X}$ as a lattice path in $\mathcal{H} = \mathbb{R}^{|\mathcal{X}|}$, that is the string $x = (a_{i_1}, \dots, a_{i_l})$ is identified as the lattice path in the vector space $\mathbb{R}^{|\mathcal{X}|}$ for which we identify $\mathcal{X}$ as an orthnomal basis (the free vector space over $\mathcal{X}$),*

$$x \in C^1([0,l], \mathcal{H}) \quad x(t) = \{t\} \cdot a_{i_{\lfloor t \rfloor}} + \sum_{r=1}^{\lfloor t \rfloor} a_{i_r},$$

where $\lfloor t \rfloor$ is the floor function of t, and $\{t\}$ is the fractional part of t. We deal with "non-geometric rough paths'', see Remark 4.5.

5.2. The global alignment kernel

The global alignment kernel is one of the most used kernels for sequences. We give its definition in modern terminology (Section 2.2 of Cuturi (2011)).

Definition 5.4 *Define*

$$\mathcal{A}(m,n) = \Big\{ \pi = (\pi_i)_{i=1}^k \in \left(\mathbb{N}^2\right)^+ : k \le m+n-1, \pi_1 = (1,1), \pi_k = (m,n), \\ (\pi_{i+1} - \pi_i) \in \{(0,1),(1,0),(1,1)\} \Big\}$$

We call an element of $\mathcal{A}(m,n)$ an alignment of (m,n).

Definition 5.5 *For a set $\mathcal{X}$ and a kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the global alignment kernel is defined as*

$$\mathrm{k}_{\mathrm{GA}} : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R} \quad (x,y) \mapsto \sum_{\pi \in \mathcal{A}(|x|,|y|)} e^{-D_{x,y}(\pi)}$$

where $D_{x,y}(\pi) = \sum_{i=1}^{|\pi|} \mathrm{k}(x_{\pi_i^1}, y_{\pi_i^2})$, and we denote with $|x|, |y|$ the length of the sequences x, y and we denote the entries of an alignment $\pi = (\pi_1, \dots, \pi_k)$ as $\pi_i = (\pi_i^1, \pi_i^2) \in \mathbb{N}^2$.

In its native form, the global alignment kernel cannot be written as a discretized signature kernel since these preserve positive definiteness, while the global alignment kernel does not (see the discussion in Cuturi (2011)). However, a simple modification turns the global alignment kernel into a positive definite kernel that can be obtained by sequentialization.

Definition 5.6 *Define*

$$\mathcal{A}_{\frac{1}{2}}(l) = \{\pi = (\pi_1, \dots, \pi_k) \in \left(\mathbb{N}^1\right)^+ : \pi_k \le l,\ \pi_{i+1} - \pi_i \in \{0,1\}\}$$

and call an element of $\mathcal{A}_{\frac{1}{2}} := \bigcup_{l \ge 1} \mathcal{A}_{\frac{1}{2}}(l)$ a half-alignment.

Definition 5.7 *Fix a set $\mathcal{X}$ and a kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The global half-alignment kernel is defined as*

$$\mathrm{k}_{\mathrm{GHA}} : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R} \quad (x,y) \mapsto \sum_{\substack{\pi \in \mathcal{A}_{\frac{1}{2}}(|x|) \\ \rho \in \mathcal{A}_{\frac{1}{2}}(|y|) \\ |\pi| = |\rho|}} e^{-D_{x,y}(\pi,\rho)}$$

where $D_{x,y}(\pi,\rho) = \sum_{i=1}^{|\pi|} \mathrm{k}(x_{\pi_i}, y_{\rho_i})$.

Proposition 5.8 *Fix a set $\mathcal{X}$ and a positive definite kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then the global half-alignment kernel $\mathrm{k}_{\mathrm{GHA}}$ equals the κ-sequentialization of the kernel $(x, y) \mapsto e^{-\mathrm{k}(x,y)}$ given in Remark 4.10(iii), that is $\mathrm{k}_{\mathrm{GHA}}(x, y) = \mathrm{k}^+(x, y)$ with $\kappa(\pi, \rho) = 1_{\pi \in \mathcal{A}_{\frac{1}{2}}} 1_{\rho \in \mathcal{A}_{\frac{1}{2}}}$.*

Proof The κ-sequentializtion yields

$$\begin{aligned}
\mathrm{k}^+(x, y) &= \sum_{\substack{m \geq 1 \\ 1 \leq \pi_1 \leq \cdots \leq \pi_m \leq |x| \\ 1 \leq \rho_1 \leq \cdots \leq \rho_m \leq |y|}} \kappa((\pi_1, \ldots, \pi_m), (\rho_1, \ldots, \rho_m)) \prod_{r=1}^{m} \mathrm{k}(x_{\pi_r}, y_{\rho_r}) \\
&= \sum_{m \geq 1} \sum_{\substack{1 \leq \pi_1 \leq \cdots \leq \pi_m \leq |x| \\ 1 \leq \rho_1 \leq \cdots \leq \rho_m \leq |y|}} 1_{(\pi_1, \ldots, \pi_m) \in \mathcal{A}_{\frac{1}{2}}} 1_{(\rho_1, \ldots, \rho_m) \in \mathcal{A}_{\frac{1}{2}}} \prod_{r=1}^{m} e^{-\mathrm{k}(x_{\pi_r}, y_{\rho_r})} \\
&= \sum_{m \geq 1} \sum_{\substack{\pi \in \mathcal{A}_{\frac{1}{2}}(|x|) \\ \rho \in \mathcal{A}_{\frac{1}{2}}(|y|) \\ |\pi| = |\rho| = m}} e^{-\sum_{i=1}^{m} \mathrm{k}(x_{\pi_i}, y_{\rho_i})} = \mathrm{k}_{\mathrm{GHA}}(x, y).
\end{aligned}$$

■

Remark 5.9 *The terms missing in k_{GA}, when compared to $\mathrm{k}_{\mathrm{GHA}}$ are those arising from sequence pairs $\pi \in \left(\mathbb{N}^2\right)^+$ in which there is an increment $(0, 0)$. In view of the discussion in Section 3.2 of Cuturi (2011), these missing terms are exactly the locus of non-transitivity in the sense of Shin and Kuboyama (2008). Concerning the possible non positive definiteness of k_{GA}, one can follow Cuturi (2011) and study sufficient conditions under which the original global alignment kernel is positive definite.*

5.3. The relation-convolution kernel

Finally, we would like to point out that the discretized signature kernels are closely related to the relation-convolution kernels, see Haussler (1999), Section 2.2.

Definition 5.10 *Fix kernels $\mathrm{k}_i : \mathcal{X}_i \times \mathcal{X}_i \to \mathbb{R}$ for $i = 1, \ldots, m$. Fix a relation R between the sets $\mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ and $\mathcal{X}$, that is a subset $R \subset \mathcal{X}_1 \times \cdots \times \mathcal{X}_m \times \mathcal{X}$ and say that $(x_1, \ldots, x_m) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ decomposes $x \in \mathcal{X}$ if $(x_1, \ldots, x_m, x) \in R$. The relation-convolution kernel of $(\mathrm{k}_1, \ldots, \mathrm{k}_m)$ is defined as*

$$\mathrm{k}_{\mathrm{RC}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (x, y) \mapsto \sum_{\substack{(x_1, \ldots, x_m) : ((x_1, \ldots, x_m), x) \in R \\ (y_1, \ldots, y_m) : ((y_1, \ldots, y_m), y) \in R}} \prod_{i=1}^{m} \mathrm{k}_i(x_i, y_i).$$

The intuition for above definition is that we can measure similarity of two objects $x, y \in \mathcal{X}$ by decomposing them into substructures and comparing these substructures.

Proposition 5.11 *Let $\mathcal{X}$ be any set and $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then the sequentialization $\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$ given in Remark 4.10 (i) is a relation-convolutional kernel for the relation*

"being an ordered subsequence''. More precisely, define $\mathcal{X}_1 = \mathbb{N}$, $\mathcal{X}_i = \mathcal{X}$ *for* $i = 2, \dots, m+1$ *and*

$$\mathrm{k}_1(i,j) = 1_{i=j} \text{ and } \mathrm{k}_i(x,y) = \mathrm{k}(x,y) \text{ for } i = 2, \dots, m+1.$$

Then for

$$R = \{(l, x_1, \dots, x_{m+1}, x') \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_{m+1} \times \mathcal{X}^+ : l \le m \text{ and } \exists i_1 < \cdots < i_l \\ \text{such that } (x'_1, \dots, x'_l) = (x_{i_1}, \dots, x_{i_l})\}$$

the relation-convolutional kernel k_{RC} *equals* k^+.

The proof follows by spelling out the definitions of k_{RC} and k^+. It is interesting to note the native discretized signature kernel of Theorem 4 does not, at least naturally, fall into Haussler's convolutional kernel framework since it heavily relies on the additional differences $\nabla \mathrm{k}$. (Above variation is only obtained after the summation step described in Remark 4.10).

6. Efficient algorithms

In this section, we design effective algorithms to evaluate or approximate the $(n \times n)$-Gram matrix $\left(\mathrm{k}^+_m(x_i, x_j)\right)_{i,j\in\{1,\dots,n\}}$ of n sequences $x_1, \dots, x_n \in \mathcal{X}^+$. The results are summarized in Table 2. The key are the recursive Horner-type formula given in Corollary 4.9 together with ideas from dynamic programming and low-rank approximations.

method	algorithm	time	storage
naive evaluation	Definition 4.8	$O(n^2 \cdot \ell \cdot \dim(\mathcal{H})^m)$	$O(n^2 \dim(\mathcal{H})^m)$
dynamic programming	Algorithm 3	$O(n^2 \cdot \ell^2 \cdot m)$	$O(\ell^2)$
DP & LR, per-element	Algorithm 4	$O(n^2 \cdot \ell \cdot \rho \cdot m)$	$O(\ell \cdot \rho)$
DP & LR, per-sequence	Section 6.3.1	$O((n+\rho) \cdot \ell^2 \cdot \rho \cdot m)$	$O(n \cdot \ell^2)$
DP & LR, simultaneous	Algorithm 5	$O(n \cdot \ell \cdot \rho \cdot m)$	$O(n \cdot \ell \cdot \rho)$

Table 2: Computational time (in elementary arithmetic operations) and storage cost for computing (an approximation) to the $(n \times n)$-Gram matrix $\left(\mathrm{k}^+_m(x_i, x_j)\right)_{i,j\in\{1,\dots,n\}}$ between sequences $x_1, \dots, x_n \in \mathcal{X}^+$ of length at most ℓ, that is $|x_i| \le \ell$ for $i = 1, \dots, n$. Methods: DP = dynamic programming, LR = low-rank. In the low-rank methods, ρ is a meta-parameter which controls accuracy of approximation.

Remark 6.1 *Naive evaluation of* $\mathrm{k}^+_m(x,y)$ *by directly calculating it as an inner product of signature features incurs an exponential storage cost* $O(\dim(\mathcal{H})^m)$. *Even for* $\mathcal{H} = \mathbb{R}^d$ *this becomes quickly very expensive and infeasible when* $\dim(\mathcal{H}) = \infty$ *which is the case for most practically relevant kernels.*

Remark 6.2 *Above algorithms are already powerful for* $\mathcal{H} = \mathbb{R}^d$ *with the inner product in* $\mathbb{R}^d$ *as kernel* k. *In this case,* $\mathrm{k}^+_x \approx \mathrm{S}(\mathrm{k}_x) = \mathrm{S}(\langle \mathrm{k}_x, \cdot \rangle) \simeq \mathrm{S}(x) \in \prod_{m\ge 0} \left(\mathbb{R}^d\right)^{\otimes m}$, *that is the feature map for a path* x *is the usual signature* $\mathrm{S}(x) \in \prod_{m\ge 0} \left(\mathbb{R}^d\right)^{\otimes m}$. *Compared to naive computation of* $\mathrm{S}(x)$, *Algorithm 5 gives a massive reduction in complexity (from exponential to linear!).*

Remark 6.3 *For readability we have absorbed in Table 2 the computational time c that is needed to evaluate $\mathrm{k}(x,y)$ for $x,y \in \mathcal{X}$ into the big O bound since this is just a multiplicative factor. E.g. the computational time for Algorithm 3 would read $O(c \cdot n^2 \cdot \ell^2 \cdot m)$ if we include c.*

6.1. Notation and subroutines for computations with arrays (tensors)

We introduce notation (inspired by Python/Numpy) and fast algorithms for dynamic programming subroutines. This allows us to express our algorithms as operations on arrays alone and makes it easy to implement them in high-level languages that are optimized for operating on arrays (such as Numpy, etc).

Notation 6.4 (Arrays/tensors) *We will denote the $(i_1,\dots,i_k)$-th element of a k-fold array (= degree k tensor) A by $A[i_1,\dots,i_k]$. Occasionally, for ease of reading, we will use "$|$" instead of "," as a separator, for example $A[i_1,i_2|i_3,\dots,i_k]$ if the indices on the left side of "$|$" are semantically distinct from those on the right side. The arrays all contain elements in $\mathbb{R}$, and the indices will always be positive integers, excluding zero.*

Notation 6.5 (functions applied elementwise) *For a function $f:\mathbb{R}\to\mathbb{R}$, and an array A, we will denote by $f(A)$ the array where f is applied element-wise. I.e., $f(A)[i_1,\dots,i_k] = f(A[i_1,\dots,i_k])$. Similarly, for $f:\mathbb{R}^m\to\mathbb{R}$ and arrays $A_1,\dots,A_m$, we denote*

$$f(A_1,\dots,A_m)[i_1,\dots,i_k] = f(A_1[i_1,\dots,i_k],\dots,A_m[i_1,\dots,i_k]).$$

For example, $\frac{1}{2}\cdot A^2$ is the array A having all elements squared, then divided by two. The array $A+B$ contains, element-wise, sums of elements of A and B.

Notation 6.6 (slice, shift, cumulative sum) *Let A be a k-fold array of size $(n_1\times\cdots\times n_k)$.*

(i) For an index i_j (at j-th position), we will write $A[:,\dots,:,i_j,:,\dots,:]$ for $(k-1)$-fold array of size $(n_1\times\cdots\times n_{j-1}\times n_{j+1}\times\dots n_k)$ such that

$$A[:,\dots,:,i_j,:,\dots,:][i_1,\dots,i_{j-1},i_{j+1},\dots,i_k] = A[i_1,\dots,i_k].$$

We define in analogy, iteratively, $A[:,\dots,:,i_j,:,\dots,:,i_{j'},:,\dots,:]$, and so on. Arrays of this type are called slices (of A).

(ii) For an integer m, we will write $A[:,\dots,:,+m,:,\dots,:]$ for the k-fold array of size $(n_1\times\cdots\times n_{j-1}\times(n_j+m)\times n_{j+1}\times\cdots\times n_k)$ such that

$$A[:,\dots,:,i_j,:,\dots,:][i_1,\dots,i_{j-1},i_j+m,i_{j+1},\dots,i_k] = A[i_1,\dots,i_k],$$

and where non-existing indices of A are treated as zero. Arrays of this type are called shifted (versions of A). For negative m, the shifts will be denoted with a "minus"-sign instead of a "plus"-sign.

(iii) We will write $A[:,\dots,:,\boxplus,:,\dots,:]$*, where* $\boxplus$ *is at the* j*-th position, for the* k*-fold array of size* $(n_1 \times \cdots \times n_k)$ *such that*

$$A[:,\dots,:,\boxplus,:,\dots,:][i_1,\dots,i_k] = \sum_{\kappa=i_j}^{n_j} A[i_1,\dots,i_{j-1},\kappa,i_{j+1},\dots,i_k].$$

Arrays of this type are called slice-wise cumulative sums (of A*).*

(iv) We will write $A[:,\dots,:,\Sigma,:,\dots,:]$*, where* Σ *is at the* j*-th position, for the* $(k-1)$*-fold array of size* $(n_1 \times \cdots \times n_{j-1} \times n_{j+1} \times \cdots \times n_k)$ *such that* $A[:,\dots,:,\Sigma,:,\dots,:][i_1,\dots,i_{k-1}] = \sum_{\kappa=1}^{n_j} A[i_1,\dots,i_{j-1},\kappa,i_j,\dots,i_{k-1}]$*. Arrays of this type are called slice-wise sums (of* A*).*

We will further use iterations and mixtures of the above notation, noting that the index-wise sub-setting, shifting, and accumulation commute with each other. Therefore expressions such as $A[+1,:|\Sigma,+2]$ *or* $A[j|:,+3,\boxplus]$ *are well-defined, for example. We will also use the notation* $A[\boxplus+m,\dots]$ *to indicate the shifted variant of the cumulative sum array.*

Cumulative sums can be computed efficiently, in the order of the size of an array, as opposed to squared complexity of more naive approaches. The algorithm is classical, we present it for the convenience of the reader in Algorithms 1 and 2.

Algorithm 1 Computing the cumulative sum of a vector.
Input: A 1-fold array A of size (n)
Output: The cumulative sum array $A[\boxplus]$

1: Let $Q \leftarrow A$.
2: **for** $\kappa = 2$ to n **do**
3: $\quad Q[\kappa] \leftarrow Q[\kappa-1] + A[\kappa]$
4: **end for**
5: Return Q

Algorithm 2 Computing the cumulative sum of an array.
Input: A k-fold array A of size $(n_1 \times \cdots \times n_k)$
Output: The cumulative sum array $A[\boxplus,\dots,\boxplus,:,\dots,:]$ (up to the m-th index)

1: Let $Q \leftarrow A$
2: **for** $\kappa = 2$ to m **do**
3: $\quad$ Let $Q \leftarrow Q[:,\dots,:,\boxplus,:,\dots,:]$ (at the κ-th index), where the right side is computed via applying algorithm 1 slice-wise.
4: **end for**
5: Return Q

6.2. Computing the discretized signature kernel

Algorithm 3 Evaluation of k_m^+.
Input: Sequences $x, y \in \mathcal{X}^+$. A kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A truncation degree m.
Output: $\mathrm{k}_m^+(x, y)$

1: Set $\ell_x = |x| - 1, \ell_y = |y| - 1$
2: Compute the $(\ell_x \times \ell_y)$-array K with entries $K[i, j] = \nabla_{i,j} \mathrm{k}(x, y)$
3: Initialize an $(m \times \ell_x \times \ell_y)$-array A
4: Set $A[1| :, :] \leftarrow K$
5: **for** $d = 2$ to m **do**
6: Compute $Q \leftarrow A[d - 1|\boxplus, \boxplus]$
7: Set $A[d| :, :] \leftarrow K \cdot (1 + Q[+1, +1])$
8: **end for**
9: Compute $R \leftarrow 1 + A[m|\Sigma, \Sigma]$
10: Return R

The correctness of above Algorithm follows directly from the recursive formula (3) in Corollary 4.9: we start by evaluating the innermost parenthesis and then use the summation to build the content of the next parenthesis. In symbols, define for $d = 1, \dots, m - 1$,

$$A^1_{i,j} := \nabla_{i,j} \mathrm{k}(x, y) \text{ and } A^{d+1}_{i,j} := \nabla_{i,j} \mathrm{k}(x, y) \left(1 + \sum_{\substack{i'>i \\ j'>j}} A^d_{i',j'} \right),$$

then $\mathrm{k}_m^+(x, y) = 1 + \sum_{\substack{i \geq 1 \\ j \geq 1}} A^m_{i',j'}$. In above Algorithm 3 we denote $A[d|i, j] = A^d_{i,j}$.

Remark 6.7

- *Disregarding the cost of computing the $(\ell_x \times \ell_y)$-matrix $(\nabla_{i,j} \mathrm{k}(x, y))_{i,j}$, the computational cost of Algorithm 3 is $O(m|x||y|)$ elementary arithmetic operations (= the number of loop elements) and $O(m|x||y|)$ units of elementary storage. The storage requirement can be reduced to $O(|x||y|)$ by discarding $A[d - 1| :, :]$ from memory after step 7 each time.*

- *In each run of the loop, a matrix Q is pre-computed to avoid a five-fold loop that would be necessary with the more naive version of line 7,*

$$A[m|i, j] \leftarrow A[m|i, j] \cdot \left(1 + \sum_{i'>i} \sum_{j'>j} A[m - 1|i', j'] \right),$$

that leads to a blown up computational cost of $O(m\ell_x^2\ell_y^2)$ at the asymptotically insignificant gain of storing one $(\ell_x \times \ell_y)$-matrix less (the matrix Q).

6.3. Large scale strategies

By using Algorithm 3 one can compute the Gram matrix $\left(\mathrm{k}_m^+(x_i, x_j)\right)_{i,j=1,\dots,n}$ of n sequences $x_1, \dots, x_n \in \mathcal{X}^+$ in $O(n^2 \cdot \ell^2 \cdot m)$ elementary arithmetic operations. While this is for moderate sizes of n,m and ℓ achievable on contemporary desktop computers, it becomes quickly prohibitive for large n, ℓ due to the quadratic growth — especially when combined with parameter tuning or cross-validation schemes (as later in our experiments). We present two low-rank techniques to address this: the first is classic and reduces the quadratic cost in n to linear, the second reduces the quadratic cost in ℓ to linear. Finally, we combine these two techniques to get $O(n \cdot \ell \cdot m \cdot \rho)$ computation time with ρ denoting an approximation parameter.

6.3.1. Low-rank methods for the sequence-vs-sequence kernel matrix

Since $\left(\mathrm{k}_m^+(x_i, x_j)\right)_{i,j=1,\dots,n}$ is a Gram-matrix, it is directly amenable to large-scale variants of low-rank type. Strategies of this kind include the incomplete Cholesky decomposition, Nyström approximation, or the inducing point formalism in a Gaussian process framework. For n sequences, all strategies mentioned above require evaluation of at most an $(n \times r)$ and an $(r \times r)$ matrix (where r is a meta-parameter), which costs $O((r+n) \cdot r \cdot \ell^2 \cdot m)$ elementary operations and $O((r+n) \cdot r \cdot \ell^2)$ storage[11]. Any of these low-rank strategies reduces the complexity of the Gram matrix to be linear in n, but not in ℓ since the strategy is completely independent of how the discretized signature kernel was evaluated at a given pair $(x_i, x_j) \in \mathcal{X}^+ \times \mathcal{X}^+$. For the same reason, any improvements on the cost of evaluating $\mathrm{k}_m^+(x_i, x_j)$ will combine with this strategy.

6.3.2. Low-rank methods for the element-vs-element kernel matrix

The evaluation $\mathrm{k}_m^+(x_i, x_j)$ for a pair $(x_i, x_j) \in \mathcal{X}^+$ is given by nested partial summations and multiplications applied to the $(\ell_j \times \ell_j)$-matrix $\left(\mathrm{k}(x_{i,r}, x_{j,s})\right)_{r,s}$ where $r = 1, \dots, \ell_i$, $s = 1, \dots, \ell_j$ and we denote $x_i = (x_{i,1}, \dots, x_{i,\ell_i}), x_j = (x_{j,1}, \dots, x_{j,\ell_j}) \in \mathcal{X}^+$. The low-rank methods of the previous paragraph cannot be applied naively to this $(\ell_i \times \ell_j)$- matrix: firstly, this matrix is in general non-symmetric; secondly, the summation- and multiplication-type operations require in naive form at each recursion step access to the full $(\ell_i \times \ell_j)$-matrix. The first issue is easily addressed by replacing the respective symmetric decomposition with the analogous non-symmetric one. The second issue is much harder to deal with, but below we show that by working exclusively on low-rank factorizations in each recursion step this issue can also be dealt with.

Definition 6.8 *Let A be an $(a \times b)$-matrix. For U an $(a \times r)$-matrix, and V a $(b \times r)$-matrix, we say that (U, V) is a low-rank presentation of A, of rank r, if*

$$A_{i,j} = \sum_{r'=1}^{r} U_{i,r'} V_{j,r'}$$

With $\bullet$ denoting the usual matrix multiplication this is equivalent to $A = U \bullet V^\top$.

11. followed by a slightly modified variant of the learning algorithm itself which usually costs $O((r+n) \cdot r^2)$ elementary operations and storage (at most) instead of the unmodified variant which usually costs $O(n^3)$ (at most).

The fact that A has a low-rank presentation of rank r does imply that A is of rank r or less (by equivalence of matrix rank with decomposition rank), but it does not imply that A is of rank exactly r.

We state a number of straightforward but computationally useful Lemmas that show how low-rank matrices behave under shifting, summation and cumulative summation, componentwise addition and multiplication. This allows us to replace these operations on matrices in Algorithm 4 by the operations on low-rank matrices.

Lemma 6.9 (Low-rank under cumulative summations and shifts) *Let (U, V) be a low-rank representation of rank r for an $(a \times b)$-matrix A. Then*

(i) Define A' and A'' as $A'_{i,j} = \sum_{i' \geq i} A_{i',j}$ and $A''_{i,j} = \sum_{j' \geq j} A_{i,j'}$. Define U' as $U'_{i,r} = \sum_{i' \geq i} U_{i',r}$ and V' as $V'_{j,r} = \sum_{j' \geq j} V_{j',r}$. Then (U', V) is a low-rank representation or rank r of A' of rank and (U, V') is a low-rank representation of rank r of A'.

(ii) Define A' and A'' as $A'_{i,j} = A_{i+s,j}$ and $A''_{i,j} = A_{i,j+s}$ for $s \geq 1$. Define U' and V' as $U'_{i,r} = U_{i+s,r} 1_{i+s \leq a}$ and $V'_{j,r} = V_{j+s,r} 1_{j+s \leq b}$. Then (U', V) is a low-rank representation of A' of rank r and (U, V') is a low rank representation of rank r of A''.

Proof This follows by spelling out the definition: for (i)

$$A'_{i,j} = \sum_{i' \geq i} A_{i',j} = \sum_{i' \geq i} \sum_{r'=1}^{r} U_{i',r'} V_{j,r'} = \sum_{r'=1}^{r} U'_{i,r'} V_{j,r'};$$

for (ii)

$$A'_{i,j} = A_{i+s,j} = \sum_{r'=1}^{r} U_{i+s,r'} V_{j,r'} = \sum_{r'=1}^{r} U'_{i,r'} V_{j,r'}.$$

The statements for A'' follow similarly. ■

Lemma 6.10 (Low rank under addition and multiplication) *Let A_1, A_2 be $(a \times b)$-matrices such that (U, V) is a low-rank representation of A_1 of rank r_1, and (R, S) is a low-rank representation of A_2 of rank r_2. Then*

(i) (U', V') is a low rank representation of $A_1 + A_2$ of rank $r_1 + r_2$ where

$$U'_{i,r} = \begin{cases} U_{i,r} & \text{, if } 1 \leq r \leq r_1 \\ R_{i,r-r_1} & \text{, if } 1 + r_1 \leq r \leq r_1 + r_2 \end{cases}, \; V'_{j,r} = \begin{cases} V_{j,r} & \text{, if } 1 \leq r \leq r_1 \\ S_{j,r-r_1} & \text{, if } 1 + r_1 \leq r \leq r_1 + r_2 \end{cases}$$

(ii) $(\bar{U}, \bar{V})$ is a low rank representation of[12] $A_1 \circ A_2$ of rank $r_1 r_2$ where

$$\bar{U}_{i,r} = U_{i,a} R_{i,b} \text{ and } \bar{V}_{j,r} = V_{j,a} S_{j,b}$$

and a, b are such that $r = (a-1) \cdot r_2 + b - 1$ with $a \in \{1, \ldots, r_1\}$ and $b \in \{1, \ldots, r_2\}$

12. We denote the elementwise product of matrices with $A \circ B$, that is $(A \circ B)_{i,j} = A_{i,j} B_{i,j}$.

Proof Since the (i,j)-entry of the matrix A_1+A_2 equals $\sum_{r=1}^{r_1} U_{i,r}V_{j,r}+\sum_{r'=1}^{r_2} R_{i,r'}S_{j,r'}$ and the (i,j)-entry of $A_1 \circ A_2$ equals $\sum_{r=1}^{r_1} U_{i,r}V_{j,r} \sum_{r'=1}^{r_2} R_{i,r'}S_{j,r'}$, the result follows by substituting the two sums over $r \in \{1,\dots,r_1\}$ and $r' \in \{1,\dots,r_2\}$ by one sum over $\{1,\dots,r_1+r_2\}$ resp. over $\{1,\dots,r_1 r_2\}$. ■

In view of Point (ii) of above Lemma we introduce new array notation:

Notation 6.11 *Let A_1 be a k-fold array of size $(n_1 \times \cdots \times n_k)$ and A_2 be a k-fold array of size $(n_1 \times \cdots \times n_{k-1} \times n'_k)$. Write $A_1 \star A_2$ for the $(n_1 \times \cdots \times n_{k-1} \times (n_k \cdot n'_k))$ array with entries*

$$A_1 \star A_2[i_1,\dots,i_k] = A_1[i_1,\dots,i_{k-1},a]A_2[i_1,\dots,i_{k-1},b]$$

where a,b are such that $i_k = (a-1)\cdot n_2 + b - 1$ with $a \in \{1,\dots,n_1\}$ and $b \in \{1,\dots,n_2\}$.

Algorithm 4 Evaluation of k_m^+, with low-rank speed-up.
Input: Sequences $x, y \in \mathcal{X}^+$. A kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A truncation degree $m \geq 1$. An integer $1 \leq r \leq \min(\ell_x, \ell_y)$.
Output: An approximation to $\mathrm{k}_m^+(x,y)$

1: Set $\ell_x \leftarrow |x| - 1$ and $\ell_y \leftarrow |y| - 1$
2: Compute a low-rank representation (U,V) of rank r, approximating the element-vs-element matrix/array with entries $K[i,j] = \nabla_{i,j}\,\mathrm{k}(x,y)$, $i \in \{1,\dots,\ell_x\}$, $j \in \{1,\dots,\ell_y\}$
3: Initialize an $(m \times \ell_x \times *)$-array B and an $(m \times \ell_y \times *)$-array C (* means that the size may change dynamically)
4: Set $B[1|\,:,:] \leftarrow U$ and $C[1|\,:,:] \leftarrow V$
5: **for** $d = 2$ to m **do**
6: $\quad P \leftarrow B[d-1| \boxplus +1,:]$ and $Q \leftarrow C[d-1| \boxplus +1,:]$
7: $\quad$ Append an $(\ell_x \times 1)$-array of 1's to P and append an $(\ell_y \times 1)$-array of 1's to Q
8: $\quad$ Set $B[d|\,:,:] \leftarrow (U \star P)[:,:]$ and $C[d|\,:,:] \leftarrow (V \star Q)[:,:]$
9: $\quad$ optional: "simplify" the low-rank presentation (B,C), reducing its rank
10: **end for**
11: Set $R \leftarrow B[m|\Sigma,:]$ and $S \leftarrow C[m|\Sigma,:]$
12: Return $1 + (R \cdot S)[\Sigma]$

Algorithm 4 first approximates the element-vs-element $(\ell_x \times \ell_y)$-matrix $K = (\nabla_{i,j}\,\mathrm{k}(x,y))_{i,j}$ by a low rank approximation (U,V). It then uses the formula for k_m^+ given in Corollary 4.9 and updates the low-rank representation in each recursion step (starting from the innermost parenthesis):

- The first time line 6 is called it uses Lemma 6.9 to calculate a low-rank representation (P,Q) of the $(\ell_x \times \ell_y)$-matrix $\left(\sum_{i'>i,j'>j} \nabla_{i',j'}\,\mathrm{k}(x,y)\right)_{i,j}$ from the low-rank approximation (U,V) for $K = (\nabla_{i,j}\,\mathrm{k}(x,y))_{i,j}$; later calls of line 6 do the same but with K replaced by another matrix with low-rank factorization stored in $B[d|\,:,:]$ and $C[d|\,:,:]$.

- The first time line 7 is called, it calculates a low-rank representation $(\bar{P}, \bar{Q})$ of the $(\ell_x \times \ell_y)$-matrix $(1 + (P \bullet Q^\top)_{i,j})_{i,j}$ by using (P, Q) and the trivial identity

$$1 + \sum_{k=1}^{r} P_{i,k} Q_{j,k} = \sum_{k=1}^{r+1} \bar{P}_{i,k} \bar{Q}_{j,k}$$

  where $\bar{P}$ and $\bar{Q}$ are given by adding a $(r+1)$-th column consisting of 1's. Note that the rank increases by one by going from (P, Q) to $(\bar{P}, \bar{Q})$. Later calls do the same but with (P, Q) replaced by another matrix with low-rank factorization stored in $B[d| :, :]$ and $C[d| :, :]$.

- Line 8 calculates a low-rank representation $(\bar{U}, \bar{V})$ for the elementwise product between the $(\ell_x \times \ell_y)$-matrices $K = U \bullet V^\top$ and $\bar{P} \bullet \bar{Q}^\top$. By Lemma 6.10 it is given as $\bar{U}_{i,\bar{r}} = U_{i,a} \bar{P}_{i,b}$ and $\bar{V}_{\bar{r},j} = V_{a,j} \bar{Q}_{b,j}$.

- Line 9 is optional, aiming at keeping the low-rank presentation small[13].

- Line 11 uses that

$$\sum_{i \geq 1, j \geq 1} (R \bullet S^\top)_{i,j} = \sum_{i \geq 1, j \geq 1} \sum_{k} R_{i,k} S_{j,k} = \sum_{k} (\sum_{i \geq 1} R_{i,k})(\sum_{j \geq 1} S_{j,k}).$$

  (Recall that $R \cdot S$ denotes elementwise multiplication of arrays R, S and $R \bullet S$ denotes the usual matrix multiplication if R, S are interpreted as matrices).

- The computational cost of Algorithm 4 is of the same order as the maximum size of B and C. That is, if ρ is the smallest integer such that at any time B requires $\ell_x \cdot m \cdot \rho$ space, and C requires $\ell_y \cdot m \cdot \rho$ space, then the computational complexity of Algorithm 4 is[14] $O((\ell_x + \ell_y) \cdot \rho \cdot m)$.

6.3.3. Simultaneous low-rank methods

Algorithm 4 yields an efficient low-rank speed-up for computing a single entry $\mathrm{k}_m^+(x_i, x_j)$. Thus total computational cost for the Gram matrix $(\mathrm{k}_m^+(x_i, x_j))_{i,j \in \{1,\dots,n\}}$ is $O(n^2 \cdot \ell \cdot \rho \cdot m)$ and cost quadratic in the number of data points n may be prohibitive on large scale data. We address this by combining both low-rank strategies mentioned before to achieve a further reduction of computational cost from $O(n^2 \cdot \ell \cdot \rho \cdot m)$ to $O(n \cdot \ell \cdot \rho \cdot m)$.

13. This can be achieved for example via singular value decomposition, sub-sampling, or random projection type techniques.
14. Since one can always choose a low-rank representation of $B[m| :, :]$ and $C[m| :, :]$ of rank $\min(\ell_x, \ell_y)$ or less, the computational complexity is bounded by $O(\ell_x \cdot \ell_y \cdot m)$ (which is the complexity of Algorithm 3).

Algorithm 5 Computation of the Gram matrix of k_m^+, with (double) low-rank speed-up.
Input: Sequences $x_1, \dots, x_n \in \mathcal{X}^+$. A kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A truncation degree m. An integer $r \geq 1$.
Output: A low-rank approximation (U, U) of the Gram matrix $K = \left(\mathrm{k}_m^+(x_i, x_j)\right)_{i,j=1,\dots,n}$.

1: Compute arrays $U^{(i)}$ such that each pair $(U^{(i)}, U^{(j)})$ $i, j \in \{1, \dots, n\}$ is a low-rank presentation of rank r for the $((|x_i| - 1) \times (|x_j| - 1))$-matrix $K^{(ij)}$ with entries $K^{(ij)}_{a,b} = \nabla_{a,b} \mathrm{k}(x_i, x_j)$
2: Initialize an $(n \times m \times * \times *)$-array B (where * means that the sizes may change dynamically)
3: $B[i|1| :, :] \leftarrow U^{(i)}$ for all $i \in \{1, \dots, n\}$
4: **for** $d = 2$ to m **do**
5: Compute $P \leftarrow B[: |d-1| \boxplus +1, :]$
6: Set κ, ρ such that $(n \times \kappa \times \rho)$ is the size of P
7: Append an $(n \times \kappa \times 1)$-array of ones to P
8: $B[: |d| :, :] \leftarrow B[: |1| :, :] \star P[:, :, :]$
9: optional: "simplify" the low-rank presentation encoded in B, reducing its rank
10: **end for**
11: Compute $U \leftarrow B[: |m|\Sigma, :]$
12: Return U

Algorithm 5 is obtained as follows:

- The previous Algorithm 4 transforms for any pair $x_i, x_j \in \mathcal{X}^+$ a low rank representation $(U^{(ij)}, V^{(ij)})$ of the $(\ell_i \times \ell_j)$-matrix $K^{(ij)} = (\nabla_{a,b} \mathrm{k}(x_{i,a}, x_{j,b}))_{a,b}$ into a low-rank representation $(R^{(ij)}, S^{(ij)})$ such that

$$\mathrm{k}^+(x_i, x_j) = 1 + \sum_{a \geq 1, b \geq 1} \left(R^{(ij)} \bullet (S^{(ij)})^\top \right)_{a,b}.$$

  But if $U^{(ij)}$ is independent of j and if $V^{(ij)}$ is independent i, then $R^{(ij)}$ will depend by Algorithm 4 only on i and $S^{(ij)}$ will depend only on j; hence $R^{(ij)} = S^{(ji)}$. If we denote $R^{(i)} := R^{(ij)} := S^{(ji)}$, then in above Algorithm 5, $B[i, m, a, r]$ equals $R^{(i)}_{a,r}$ when the end of the for loop is reached, line 10.

- Write

$$\begin{aligned} \mathrm{k}_m^+(x_i, x_j) &= 1 + \sum_{\substack{a \geq 1 \\ b \geq 1}} \left(R^{(i)} \bullet (R^{(j)})^\top \right)_{a,b} = 1 + \sum_{k=1}^{r'} \left(\sum_{a \geq 1} R^{(i)}_{a,k} \sum_{b \geq 1} R^{(j)}_{b,k} \right) \\ &= \sum_{k=1}^{r'+1} \bar{R}_{i,k} \bar{R}_{j,k} \end{aligned}$$

  where we denote with r' the rank of $(R^{(i)}, R^{(j)})$ and set $\bar{R}_{i,k} := \sum_{a \geq 1} R^{(i)}_{a,k}$ and $\bar{R}_{a,k} := 1$ if $k = r' + 1$. Hence, $(\bar{R}, \bar{R})$ is a low-rank representation of the $(n \times n)$-Gram matrix $K = (\mathrm{k}_m^+(x_i, x_j))_{i,j}$. In Algorithm 5, the array entry $U[i, r']$ equals $\bar{R}_{i,r'}$, line 11.

Remark 6.12 *Line 1 requires that for each pair of sequences* $x_i = (x_{i,a})_{a=1,\dots,\ell_i}, x_j = (x_{j,b})_{b=1,\dots,\ell_j} \in \mathcal{X}^+$ *we compute the* $(\ell_i \times \ell_j)$*-matrix* $K^{(ij)}$ *with entries* $K^{(ij)}_{a,b} = \mathrm{k}(x_{i,a}, x_{j,b})$*. That is, the matrices* $U^{(i)}$*, when row-concatenated, should have low rank*[15]*. Jointly low-rank* $U^{(i)}$ *can be obtained by running a suitable joint diagonalization or singular value decomposition scheme on the matrices* $K^{(ij)}$*.*

Remark 6.13 (Fast sequential kernel methods) *By Section 5, fast string kernel methods such as the gappy, substitution, or mismatch kernels presented in Leslie and Kuang (2004) may be transferred to general sequential kernels. In general, this amounts to small modification of Algorithm 3; for example, to obtain a gappy variant of the sequential kernel, summation in line 6 of Algorithm 3 over the whole matrix, of quadratic size, is replaced by summation over a linear part of it.*

7. Experimental validation

We perform two experiments to validate the practical usefulness of the signature kernels:

(1) On a real world data set of hand movement classification (eponymous UCI data set Sapsanis et al. (2013)), we show the discretized signature kernel outperforms the best previously reported predictive performance Sapsanis et al. (2013), as well as non-sequential kernel and aggregate baselines.

(2) On a real world data set on hand written digit recognition (pendigits), we show that the discretized signature kernel over the Euclidean kernel (= linear use of signature features) achieves only sub-baseline performance. Using the discretized signature kernel over a Gaussian kernel improves prediction accuracy to the baseline region.

We emphasize that our experiments do not constitute a systematic benchmark comparison to prior work, only validation that the signature kernel is a practically meaningful concept: experiment (1), shows that the sequentialization of standard kernels can achieve state-of-the-art performance on time series data; experiment (2) shows that even for paths in low dimensions, $\mathcal{X} = \mathbb{R}^2$, using a non-linear static kernel for the sequentialization outperforms the linear kernel (the latter corresponds to learning with signature features of a 2-dimensional path; the former, corresponds to learning with signature features of the path lifted to the RKHS $\mathcal{H}$ of the non-linear kernel).

A systematic benchmark comparison is likely to require a larger amount of work, since it would have to include a number of previous methods (multiple variants of the string and general alignment kernels, dynamic time warping, naive use of signatures), for most of which there is no freely available code with interface to a machine learning toolbox, and benchmark methods (order-agnostic baselines such as summary aggregation and chunking; distributional regression; naive baselines).

15. For example, if k is the Euclidean scalar product, $U^{(i)}$ can be taken as the raw data matrix, where rows are different time points and columns are features.

7.1. Validation and prediction set-up

7.1.1. Prediction tasks

In all data sets, samples are multi-variate (time) series. All learning tasks are supervised classification tasks of predicting class labels attached to series of equal length.

7.1.2. Prediction methods

For prediction, we use eps-support vector classification (as available in the python/scikit-learn package) on the kernel matrices obtained from the following kernels:

(1.a) the Euclidean kernel $\mathrm{k}(x, y) = \langle x, y \rangle$. This kernel has no parameters.

(1.b) the Gaussian kernel $\mathrm{k}(x, y) = \exp\left(\frac{1}{2}\gamma^2 \|x - y\|^2\right)$. This kernel has one parameter, a scaling constant γ.

(2.a) the (truncated) discretized signature kernel k_m^+ over the linear/Euclidean kernel $\mathrm{k}(x, y) = \gamma\langle x, y \rangle$. This kernel has two parameters, a scaling constant γ, and the truncation degree m.

(2.b) the (truncated) discretized signature k_m^+ over the Gaussian kernel $\mathrm{k}(x, y) = \theta \exp\left(\frac{1}{2}\gamma^2 \|x - y\|^2\right)$. This kernel has three parameters: scaling constants γ and θ, and truncation degree m.

(1.a) and (1.b) are considered standard kernels, (2.a) and (2.b) are discretized signature kernels. Note that the kernels (1.a) and (1.b) can only be applied to sequential data samples of equal length which is the case for the data sets considered. Even though (1.a), (1.b) may be applied to sequences of same length, they do not use any information about their ordering: both the Euclidean and the Gaussian kernel are invariant under (joint) permutation of the order of the indexing in the arguments. Another subtlety is that the discretized signature kernels (2.a), (2.b) do use information about the ordering of the sequences, but only for a truncation $m \geq 2$. For $m = 1$, the kernel corresponds to choosing the increment/mean aggregate feature (Euclidean) or a type of distributional classification (Gaussian). We will therefore explicitly compare truncation degrees 1 versus 2 and higher, to enable us to make a statement about whether using the order information was beneficial (or not).

7.1.3. Tuning and error estimation

In all experiments, we use nested (double) cross-validation for parameter tuning (inner loop) and estimation of error metrics (outer loop). In both instances of cross-validation, we perform uniform 5-fold cross-validation.

Unless stated otherwise, parameters are tuned on the tuning grid given in Table 3 (when applicable). Kernel parameters are the same as in the above section "prediction mehods". The best parameter is selected by 5-fold cross-validation, as the parameter yielding the minimum test-f1-score, averaged over the five folds.

parameter	range
kernel param. γ	0.01, 0.1, 1
kernel param. θ	0.01, 0.1, 1
truncation degree m	1,2,3
SVC regularizer	0.1, 1, 10, 100, 1000

Table 3: Tuning grid

7.1.4. Error metrics

The out-of-sample classification error is reported as precision, recall, and f1-score of out-of-sample prediction on the test fold. Errors measures are aggregated with equal weights on classes and folds. These aggregates are reported in the result tables.

7.2. Experiment: Classifying hand movements

We performed classification with the eps-support vector machine (SVC) on the hand movements data set from UCI Sapsanis et al. (2013). The first database in the data set which we considered for this experiment contains, for each of five subjects (two male, three female) 180 samples of hand movement sEMG recordings. Each sample is a time series in two variables (channels) at 3.000 time points. The time series fall into six classes of distinct types of hand movement (spherical, tip, palmar, lateral, cylindrical, hook). For each subject, 30 samples of each class were recorded. Hence, for each subject, there is a total of 180 sequences in $\mathcal{X}^{3000}$ with $\mathcal{X} = \mathbb{R}^2$.

For each of the five subjects, we conducted the classification experiment as described in Section 7.1, comparing prediction via SVC using one of the following kernels: (1.a) the Euclidean kernel, (1.b) the Gaussian kernel, (2.a) the sequentialized Euclidean kernel. For the non-sequential kernels (1.a), (1.b), prediction was performed with and without prior standardization of the data. For the sequential kernel, the tuning grid was considered in two parts: a truncation degree of $m = 1$, corresponding to mean aggregation, and truncation degrees of $m = 2, 3$, corresponding to the case where genuine sequence information is used. Further, we use the low-rank speed, Algorithm 5, to compute the Gram matrix.

The results are reported in Tables 4 to 8. Jackknife standard errors (pooling the five folds) are all 0.04 or smaller. Baseline performance of an uninformed estimator is $1/6 \approx 0.17$.

One can observe that for all five subjects, SVC with sequential kernel of degree 2 or higher

method	precision	recall	f1-score
(1.a) linear	0.37	0.38	0.36
(1.a) linear, standardized	0.33	0.32	0.29
(1.b) Gaussian	0.57	0.59	0.56
(1.b) Gaussian, standardized	0.54	0.50	0.50
(2.a) mean aggregation	0.19	0.20	0.18
(2.a) sequential, degree ≥ 2	0.87	0.86	0.86

Table 4: female1.mat

outperforms SVC using any of the other kernels not using any sequence information. The sequence kernel outperforms the reported methods from the original paper Sapsanis et al. (2013) as well (Figures 11 and 12).

method	precision	recall	f1-score
(1.a) linear	0.47	0.39	0.37
(1.a) linear, standardized	0.31	0.28	0.27
(1.b) Gaussian	0.71	0.71	0.70
(1.b) Gaussian, standardized	0.59	0.58	0.56
(2.a) mean aggregation	0.18	0.20	0.18
(2.a) sequential, degree ≥ 2	0.94	0.97	0.95

Table 5: female2.mat

method	precision	recall	f1-score
(1.a) linear	0.48	0.46	0.46
(1.a) linear, standardized	0.47	0.42	0.43
(1.b) Gaussian	0.66	0.64	0.63
(1.b) Gaussian, standardized	0.54	0.51	0.50
(2.a) mean aggregation	0.26	0.23	0.20
(2.a) sequential, degree ≥ 2	0.96	0.96	0.96

Table 6: female3.mat

method	precision	recall	f1-score
(1.a) linear	0.37	0.33	0.33
(1.a) linear, standardized	0.38	0.36	0.36
(1.b) Gaussian	0.59	0.57	0.57
(1.b) Gaussian, standardized	0.53	0.54	0.53
(2.a) mean aggregation	0.20	0.18	0.17
(2.a) sequential, degree ≥ 2	0.96	0.96	0.96

Table 7: male1.mat

method	precision	recall	f1-score
(1.a) linear	0.36	0.33	0.32
(1.a) linear, standardized	0.37	0.29	0.27
(1.b) Gaussian	0.72	0.71	0.70
(1.b) Gaussian, standardized	0.34	0.39	0.35
(2.a) mean aggregation	0.22	0.23	0.20
(2.a) sequential, degree ≥ 2	0.93	0.93	0.93

Table 8: male2.mat

7.3. Experiment: Pendigits

We performed classification on the pendgits data set from the UCI repository[16]. It contains 10992 samples of digits between 0 and 9 written by 44 different writers with a digital pen on a tablet. One sample consists of a pair of horizontal and vertical coordinates of sampled at 8 different time points, hence we deal with a sequence in $\mathcal{X}^8$ with $\mathcal{X} = \mathbb{R}^2$. The data set comes with a pre-specified training fold of 7494 samples, and a test fold of 3498 samples. Estimation of the prediction error is performed in this validation split, while tuning is done as described via nested 5-fold cross-validation, inside the pre-specified training fold.

We compared prediction via SVC using one of the following three kernels: (2.a) the sequentialized Euclidean kernel, and (2.b) the sequentialized Gaussian kernel. For both, the truncation degree was set to $m = 4$. The results are reported in Table 9. Jackknife standard errors (pooling the five folds) are all 0.01 or smaller. Baseline performance of an uninformed estimator is $1/10 \approx 0.10$.

16. `https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits`

method\method	precision	recall	f1-score
sequential, linear	0.91	0.90	0.89
sequential, Gaussian	0.97	0.97	0.97

Table 9: Pendigits

The quality of the SVC prediction with the sequentialzed linear kernel is comparable to results reported in Diehl (2013). It is outperformed by SVC prediction with the sequentialized Gaussian kernel. The latter performance is similar to the baseline performance of k-nearest neighbors reported in the documentation of the pendigits data set.

Acknowledgments

FK acknowledges support by the Alan Turing Institute, under EPSRC grant EP/N510129/1. HO is grateful for support by the Oxford-Man Institute of Quantitative finance and would like to thank Ilya Chevyrev for a careful proofreading and helpful remarks and discussions.

Appendix A. Signatures, and proofs of Theorem 1 and Theorem 3

A.1. Signatures

Definition A.1 *Let E be a normed space and denote with $L(\mathcal{H}, E)$ the set of continuous linear maps from $\mathcal{H}$ to E. Given $x \in C^1([0,1], \mathcal{H})$ and $y \in C^1([0,1], L(\mathcal{H}, E))$, the Riemann–Stieltjes integral of y over $[a,b] \subseteq [0,1]$ is defined as the element in E given as*

$$\int_a^b y \, \mathrm{d}x := \lim_{\mathrm{mesh}(\pi)\to 0} \sum_{i=1}^{l} y(t_i)\left(x(t_{i+1}) - x(t_i)\right)$$

where the limit is taken over all partitions $\pi = \{a \le t_1 < \cdots < t_l \le b\}$ and $\mathrm{mesh}(\pi) := \max_{i=1,\ldots,l-1} |t_{i+1} - t_i|$. We also use the shorter notation $\int y \, \mathrm{d}x$ if the integration domain is clear from the context.

See (Lyons, 2004, Theorem 1.16) for a proof of a more general result. Applying the above with E being the subspace of $\prod_{m\ge 0} \mathcal{H}^{\otimes m}$ of square summable tensors and the linear map being tensor multiplication, gives the series

$$\mathrm{S}(x) := (\mathrm{S}_m(x))_{m\ge 0} := \left(\int \mathrm{d}x^{\otimes m}\right)_{m\ge 0} \in \prod_{m\ge 0} \mathcal{H}^{\otimes m}$$

with $\int \mathrm{d}x^{\otimes 0} := 1$ and $\int \mathrm{d}x^{\otimes (m+1)} := \int \mathrm{d}x^{\otimes m}$.

A.2. Proof of Theorem 1

This is more or less a folk theorem in rough path theory, but we recall it for readers not familiar with the subject. Point (1) of Theorem 1 follows since $\mathrm{S}(h)$ is itself a solution of a linear ODE driven by h and this solution map is continuous, see Lyons (2004). Point (2) is

more involved and we refer to Hambly and Lyons (2010). For the last point, Point (3), the key insight is that the series $\mathrm{S}(h)$ behaves like monomials.

Theorem 5 (Shuffle product) *Let $h \in C^1([0,1], \mathcal{H})$, then*

$$\int \mathrm{d}h^{\otimes m} \otimes \int \mathrm{d}h^{\otimes m'} = \sum_{\sigma} \sigma \left(\int \mathrm{d}h^{\otimes (m+m')} \right).$$

Here the sum is taken over all ordered shuffles $\mathrm{OS}_{m,m'}$ which are defined as

$$\left\{ \sigma : \sigma \text{ permutation of } \left\{1, \dots, m+m'\right\}, \sigma(1) < \cdots < \sigma(m), \sigma(m+1) < \cdots < \sigma\left(m+m'\right) \right\}.$$

and $\sigma \in \mathrm{OS}_{m,m'}$ acts on $\mathcal{H}^{\otimes (m+m')}$ as $\sigma \left(e_{i_1} \otimes \cdots \otimes e_{i_{m+m'}} \right) = e_{\sigma(i_1)} \otimes \cdots \otimes e_{\sigma\left(i_{m+m'}\right)}$.

The proof of above is given in (Lyons, 2004, Theorem 2.29 (ii)). A direct consequence by applying Stone–Weierstrass (in analogy to classic monomials) is that *linear* functionals of signatures approximate *nonlinear* functions of paths aribtrary well, thus showing Point (3) of Theorem 1.

A.3. Proof of Theorem 3 and Corollaries 4.3 and 4.4

Definition A.2 *Denote with $\Delta^{\ell}_{[a,b]}$ the n-simplex over the interval $[a,b]$, that is*

$$\Delta^{\ell}_{[a,b]} = \{(t_i)_{i=1}^{\ell} : a = t_1 < \cdots < t_{\ell} = 1\},$$

and with $\Delta_{[a,b]} = \bigcup_{\ell \geq 2} \Delta^{\ell}_{[a,b]}$. If the interval $[a,b]$ is clear from the context is clear, then we just write Δ^{ℓ} and Δ.

The following notation for sequences becomes useful.

Notation A.3 *For $n \geq 1$ denote*

1. *$[n] := \{1, \dots, n\}$,*
2. *sequences in $[n]$ as $\boldsymbol{i} = (i_1, \dots, i_l) \in [n]^l$ and call $|\boldsymbol{i}| = l$ the length of the sequence $\boldsymbol{i}$,*
3. *$d(\boldsymbol{i}) := \max\{r \ : \ i_1 = i_2 = \cdots = i_r\}$ the number of repetitions of the first element in the sequence $\boldsymbol{i}$ where by convention $d(\boldsymbol{i}) = 0$ if $i_1 \neq i_2$,*
4. *$\boldsymbol{i}! := n_1! \cdots n_k!$ if $\boldsymbol{i}$ consists of $k = |\{i_1, \dots, i_l\}|$ different elements in $[n]$ and $n_1, \dots, n_k$ denote the number of times they occur in $\boldsymbol{i}$,*
5. *$\boldsymbol{i} \sqsubset [n]$ if $\boldsymbol{i} = (i_1, \dots, i_l) \in [n]^l$ and $i_1 < \cdots < i_l$,*
6. *$\boldsymbol{i} \sqsubseteq [n]$ if $\boldsymbol{i} = (i_1, \dots, i_l) \in [n]^l$ and $i_1 \leq \cdots \leq i_l$,*
7. *$\boldsymbol{i} \sqsubseteq_d [n]$ if $\boldsymbol{i} \sqsubseteq [n]$ and no element in the sequence $\boldsymbol{i}$ appears more than than d times.*

Lemma A.4 *Let $h \in C^1([a,b],\mathcal{H})$, $\pi \in \Delta^\ell_{[a,b]}$ and define $h^\pi = (h(t_i))_{i=1,\dots,\ell} \in \mathcal{H}^+$. For any sequence $\boldsymbol{i} = (i_k)_{k=1,\dots,m} \in [\ell]^m$ with $1 \le i_1 \le \cdots \le i_m \le \ell$ denote $\Delta_{\boldsymbol{i}} := \Delta \cap \bigtimes_{k=1}^{m-1}[t_{i_k}, t_{i_k+1}]$. Then*

$$\mathrm{S}_m(h) - \mathrm{S}^+_m(h^\pi) = \sum_{\boldsymbol{i}} \int_{\Delta_{\boldsymbol{i}}} \mathrm{d}h^{\otimes m},$$

where the sum is taken over all $\boldsymbol{i} = (i_k)_{k=1}^m$ with $1 \le i_1 \le \cdots \le i_m \le \ell$ that have at least one repeating index, that is $i_k = i_{k+1}$ for at least one $1 \le k \le m-1$. Moreover, each term in above sum can be bounded as follows

$$\left\| \int_{\Delta_{\boldsymbol{i}}} \mathrm{d}h^{\otimes m} \right\|_{\mathcal{H}} \le \frac{1}{\boldsymbol{i}!} \prod_{i \in \boldsymbol{i}} \|h|_{[t_i, t_{i+1}]}\|_1.$$

where we use $\boldsymbol{i}!$ as introduced in Notation A.3.

Proof We can decompose Δ into a disjoint union of $\Delta_{\boldsymbol{i}}$ over all sequences $\boldsymbol{i}$, with $1 \le i_1 \le \cdots \le i_m \le \ell$, hence

$$\mathrm{S}_m(h) = \int \mathrm{d}h^{\otimes m} = \sum_{\boldsymbol{i}} \int_{\Delta_{\boldsymbol{i}}} \mathrm{d}h^{\otimes m}.$$

Split the sum $\sum_{\boldsymbol{i}}$ into a sum over $\boldsymbol{i}$ that have an repeating element, i.e. $i_k = i_{k+1}$ for at least one k, and a sum $\sum_{\boldsymbol{i}}$ over indices that do not have repeating elements. For the latter, the integration can be done explicitly and this sum equals

$$\sum_{\boldsymbol{i}} \nabla_{i_1} h \otimes \cdots \otimes \nabla_{i_m} h$$

with our usual notation $\nabla_i h = h(t_{i+1}) - h(t_i)$. However, this is exactly $\mathrm{S}^+_m(h)$, hence the first statement follows.

For the first sum over sequences $\boldsymbol{i}$ with repeating elements, denote with $\bar{i}_1, \dots, \bar{i}_k$ the distinct indices in $\boldsymbol{i}$, and $n_1, \dots, n_k$ the total counts of their respective occurrences. Then $\Delta_{\boldsymbol{i}} = \bigtimes_{j=1}^k \bar{\Delta}_j$ with $\bar{\Delta}_j := \Delta^{n_j} \cap [t_{\bar{i}_j}, t_{\bar{i}_j+1}]^{n_j}$ and it follows that

$$\int_{\Delta_{\boldsymbol{i}}} \mathrm{d}h^{\otimes m} = \bigotimes_{j=1}^k \int_{\bar{\Delta}_j} \mathrm{d}h^{\otimes n_j}.$$

Therefore, we obtain

$$\left\| \bigotimes_{j=1}^k \int_{\bar{\Delta}_j} \mathrm{d}h^{\otimes n_j} \right\|_{\mathcal{H}} = \prod_{j=1}^k \left\| \int_{\bar{\Delta}_j} \mathrm{d}h^{\otimes n_j} \right\|_{\mathcal{H}} \le \frac{1}{\boldsymbol{i}!} \prod_{i \in \boldsymbol{i}} \|h|_{[t_i, t_{i+1}]}\|_1.$$

■

This is enough to prove our main approximation result:

Theorem 6 *Let $h \in C^1([0,1],\mathcal{H})$ let $\pi \in \Delta^\ell$. Then for every $m \ge 0$*

$$\|\, \mathrm{S}_m(h) - \mathrm{S}^+_m(h^\pi)\|_{\mathcal{H}^{\otimes m}} \le g_m$$

where (g_m) *are the coefficients in the Taylor expansion around* 0 *of the function*

$$g(z) := \sum_{m=1}^{\infty} g_m \cdot z^m := \exp(z \cdot \|h\|_1) - \prod_{i=1}^{\ell-1} \left(1 + z \cdot \|h|_{[t_i,t_{i+1}]}\|_1\right).$$

Moreover,

$$\|\,\mathrm{S}(h) - \mathrm{S}^+(h^\pi)\| \leq \exp(\|h\|_1) - \prod_{i=1}^{\ell-1} \left(1 + \|h|_{[t_i,t_{i+1}]}\|_1\right)$$

If π *is chosen such that* $\|h|_{[t_i,t_{i+1}]}\|_1 = \frac{1}{\ell-1}\|h\|_1$ *for all* $i = 1, \ldots, \ell-1$*, then*

$$\|\,\mathrm{S}(h) - \mathrm{S}^+(h)\| \leq \frac{\exp(\|h\|_1)}{\ell - 1} \left(1 + \frac{\|h\|_1^{\ell-1}}{(\ell-3)!}\right).$$

Proof The bound for $\|\,\mathrm{S}_m(h) - \mathrm{S}_m^+(h^\pi)\|_{\mathcal{H}^{\otimes m}}$ follows from Lemma A.4 by explicitly writing out the coefficient g_m. Applying the triangle inequality and using this bound and one obtains

$$\|\,\mathrm{S}(h) - \mathrm{S}^+(h^\pi)\| \leq \sum_{m=1}^{\infty} \|\,\mathrm{S}_m(h) - \mathrm{S}_m^+(h^\pi)\|_{\mathcal{H}^{\otimes m}} \leq \sum_{m=1}^{\infty} g_m = g(1)$$

Finally, for the special choice of a uniform partition π

$$g(1) = \exp\|h\|_1 - \left(1 + \frac{\|h\|_1}{\ell - 1}\right)^{\ell-1},$$

to which we apply Euler's approximation theorem, Theorem 7, below using that $\|h|_{[a,b]}\|_1 + \|h|_{[b,c]}\|_1 = \|h|_{a,c}\|_1$. ∎

The first part of Theorem 6 implies Theorem 3, the second part implies Corollary 4.4. For readers' convenience we recall Euler's well-known approximation to the exponential function.

Theorem 7 *Let* $x \in \mathbb{R}, n \in \mathbb{N}$*. Then,*

$$\left(1 + \frac{x}{n}\right)^n - \exp(x) = g(x,n), \qquad \textit{with } |g(x,n)| \leq \frac{\exp(x)}{n}\left(1 + \frac{x^n}{(n-2)!}\right).$$

In particular, it holds that

$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = \exp(x),$$

where convergence is uniform of order $O(n^{-1})$ *on any compact subset of* $\mathbb{R}$*.*

Proof All statements follow from the first, which we proceed to prove. By the binomial theorem, it holds that

$$\left(1 + \frac{x}{n}\right)^n = \sum_{k=0}^{n} \binom{n}{k} \cdot \frac{x^k}{n^k}.$$

From the definition of the binomial coefficient and an elementary computation, one obtains

$$\binom{n}{k} \cdot \frac{x^k}{n^k} = \frac{x^k}{k!} + g(x,n,k), \text{ where } \|g(x,n,k)\| \leq \frac{x^k}{k!n},$$

for $k \leq n$. For $k \geq n$, one has

$$\frac{x^k}{k!} \leq \frac{x^n}{n!} \cdot \frac{x^{k-n}}{(k-n)!}.$$

Putting together all inequalities and using the Taylor expansion of exp yields the claim. ■

To prove Corollary 4.3 we need to generalize Euler's approximation of $\exp x$ to the case when x is not divided into uniform parts, but potentially very unbalanced parts. This is done in the proposition below

Proposition A.5 *Let $x \in \mathbb{R}, n \in \mathbb{N}$, $x \geq 0$. Let $x_1, \dots, x_\ell \in \mathbb{R}$, $x_i \geq 0$ for $i = 1, \dots, \ell$ such that $\sum_{i=1}^{\ell} x_i = x$. Then,*

$$\exp(x) = \prod_{i=1}^{\ell} (1 + x_i) + g(x, x_1, \dots, x_\ell), \quad \text{where } 0 \leq g(x, x_1, \dots, x_\ell) \leq x \exp(x) \cdot \max_{i=1,\dots,\ell} x_i.$$

In particular, it holds that

$$\lim_{\max_i x_i \to 0} \prod_i (1 + x_i) = \exp(x),$$

where convergence is uniform of order $O(\max_i x_i)$ on any compact subset of $[0, \infty)$.

Proof All statements follow from the first, which we proceed to prove. We use Notation A.3 Writing out the product, we obtain

$$\prod_{i=1}^{\ell} (1 + x_i) = \sum_{\boldsymbol{i} \sqsubset [\ell]} x^{\boldsymbol{i}},$$

where abbreviatingly we have written $x^{\boldsymbol{i}} := \prod_{i \in \boldsymbol{i}} x_i$. The Taylor expansion of the exponential on the other hand yields

$$\exp(x) = \exp\left(\sum_{i=1}^{\ell} x_i\right) = \sum_{\boldsymbol{i} \sqsubseteq [\ell]} \frac{1}{\boldsymbol{i}!} x^{\boldsymbol{i}}.$$

Note the major different between both sums above being the repeating indices which may occur in the expansions of $\exp(x)$. More precisely, we obtain

$$\exp(x) - \prod_{i=1}^{m} (1 + x_i) = \sum_{\substack{\boldsymbol{i} \sqsubseteq [\ell] \\ \boldsymbol{i}! > 1}} \frac{1}{\boldsymbol{i}!} x^{\boldsymbol{i}}.$$

We further split up the sum by length of $\boldsymbol{i}$:

$$\exp(x) - \prod_{i=1}^{\ell}(1+x_i) = \sum_{m=2}^{\infty} \sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell] \\ |\boldsymbol{i}|=m \\ \boldsymbol{i}!>1}} \frac{1}{\boldsymbol{i}!}x^{\boldsymbol{i}}.$$

Positivity of g follows from this equation and positivity of x. Now consider the map ϕ which removes the first duplicated index in an ordered index sequence $\boldsymbol{i}$ yielding a sequence of length $|\boldsymbol{i}|-1$. On sequences of length m, the map ϕ is at most m-to-one, and surjective onto sequences of length $m-1$. Therefore,

$$\sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell] \\ |\boldsymbol{i}|=m \\ \boldsymbol{i}!>1}} \frac{1}{\boldsymbol{i}!}x^{\boldsymbol{i}} \leq X \cdot \sum_{m=2}^{\infty} \frac{m}{2} \sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell] \\ |\boldsymbol{i}|=m-1}} \frac{1}{\boldsymbol{i}!}x^{\boldsymbol{i}},$$

where $X = \max_i x_i$. Thus,

$$\sum_{m=2}^{\infty} \sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell] \\ |\boldsymbol{i}|=m \\ \boldsymbol{i}!>1}} \frac{1}{\boldsymbol{i}!}x^{\boldsymbol{i}} \leq X \cdot \sum_{m=1}^{\infty} m \cdot \sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell] \\ |\boldsymbol{i}|=m}} \frac{1}{\boldsymbol{i}!}x^{\boldsymbol{i}}.$$

Comparing to the expansion of $\exp(x)$ above, one observes that the right hand side is equal to

$$X \cdot \sum_{m=1}^{\infty} m \cdot \frac{x^m}{m!} = X \cdot x \sum_{m=0}^{\infty} \frac{x^m}{m!} = X \cdot x \cdot \exp(x).$$

■

The bounds in Proposition A.5 are worse than those from Euler's classic approximation result to the exponential in the case of equal x_i, by a factor of x. This is due to the fact that the bound also needs to be valid for heavily imbalanced partitions of x into x_i.

Appendix B. Higher order signature kernels and noisy observations

The discretized signature kernel was defined as inner product

$$\mathrm{k}^{+} : \mathcal{X}^{+} \times \mathcal{X}^{+} \to \mathbb{R} \quad \mathrm{k}^{+}(x,y) = \langle \mathrm{S}^{+}(\mathrm{k}_x), \mathrm{S}^{+}(\mathrm{k}_y) \rangle$$

of features S^{+} that approximate the signature S, thus k^{+} will approximate the signature kernel

$$\mathrm{k}^{\oplus} : \mathcal{P}_{\mathcal{X}} \times \mathcal{P}_{\mathcal{X}} \to \mathbb{R} \quad \mathrm{k}^{\oplus}(x,y) = \langle \mathrm{S}(\mathrm{k}_x), \mathrm{S}(\mathrm{k}_y) \rangle$$

when the sequences are discretizations of paths: in Section 4 we showed under Assumption 3.1 that

$$\mathrm{k}^{+}(x^{\pi}, y^{\pi'}) \to \mathrm{k}^{\oplus}(x,y) \text{ as } \max(\mathrm{mesh}(\pi), \mathrm{mesh}(\pi')) \to 0.$$

For essentially all practically relevant kernels $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ Assumption 3.1 holds whever the underlying paths x, y are of bounded variation. However, a common situation is that observations are perturbed by noise which are generically of unbounded variation.

Example B.1 *Let $\mathcal{X} = \mathbb{R}$ and $\mathrm{k}(x, y) = \langle x, y \rangle_{\mathbb{R}}$ (that is $\mathcal{H} = \mathbb{R}$). If $x \in C^1([0, 1], \mathcal{X})$ and $B = (B_t)_{t\in[0,1]}$ is a Brownian motion in $\mathcal{X}$ then $\mathrm{S}(\mathrm{k}_{x+B}) \simeq \mathrm{S}(x + B) = \left(\int \mathrm{d}(x + B)^{\otimes m}\right)$ is not well-defined as a Riemann–Stieltjes integral.*

A general strategy is to replace the signature map $x \mapsto \mathrm{S}(x)$ by a a map from paths to $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$ as follows: construct a sequence of bounded variation paths (x_n) that approximation x and such that $t \mapsto \left(\int_0^t \mathrm{d}x_n^{\otimes m}\right)$ converges to a $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$-valued path which is called the "*(geometric) rough path lift*" of x.

Example B.2 *Constructions for mapping a path to an element of $\prod_{m\geq 0} \mathcal{H}^{\otimes m}$ are known for a wide range of stochastic processes; below we mention a few*

- Brownian motion *(leading to p-rough paths for any $p > 2$)*
- *more generally, continuous* Semimartingale *(leading to p-rough paths for any $p > 2$),*
- fractional Brownian motion *of Hurst parameter $H > \frac{1}{4}$,*
- *more generally,* Gaussian processes *(leading to p-rough paths where p depends on the regulary of the covariation process),*
- Markov processes *in continuous time (leading to p-rough paths with p depending on the generator of the Markov process).*

For details and more examples see Lyons (2004).

The notion of a geometric rough path allows to study classes of much "rougher" paths by the same approach we developed for the bounded variation case: below we provide the needed modifications for obtaining a sequentialized kernel such that $\mathrm{k}^+(x^\pi, y^{\pi'})$ is still well-defined as the mesh vanishes when x, y are paths of unbounded variation but a "rough path lift" exists (e.g. all the stochastic processes in Example B.2)

Remark B.3 *A more radical approach is to assume that the we are not only given paths as data but also the first levels of their "rough path lift", see for example Crisan et al. (2013). The same recursion applies to give a kernel, however, we do not spell out details since all benchmark data sets we are aware of, only provide path increments.*

Definition B.4 *Let $d \in \mathbb{N}$. We call the map*

$$\mathrm{S}^+_{(d)} : \mathcal{H}^+ \to \prod_{m\geq 0} \mathcal{H}^{\otimes m}, \quad (h_i)_{i=1}^\ell \mapsto \prod_{i=1}^{\ell} \sum_{j=0}^{d} \frac{(\nabla_i h)^{\otimes j}}{j!}$$

the discretized signature map of degree d. We denote with $\mathrm{S}^+_{(d,m)}$ the projection of $\mathrm{S}^+_{(d)}$ to $\bigoplus_{n=0}^m \mathcal{H}^{\otimes n}$.

Definition B.5 *Let* $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *and* $d, m \geq 1$, $d \geq m$. *The discretized signature kernel over* k *of order* d *and at degree* m *is defined as*

$$\mathrm{k}^+_{(d,m)}(x, y) : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad \mathrm{k}^+_{(d,m)}(x, y) = \langle \mathrm{S}^+_{(d,m)}(\mathrm{k}_x), \mathrm{S}^+_{(d,m)}(\mathrm{k}_y) \rangle.$$

Remark B.6

- *The sequentialization* k^+_m *from Section 4 arises as special case of above, more general, definition:* $\mathrm{k}^+_m = \mathrm{k}^+_{(d,m)}$ *for* $d = 1$, $m \in \mathbb{N}$.
- *Readers familiar with rough paths will notice that the choice* $m = d = \lfloor p \rfloor$ *recovers the notion of* $(\mathrm{k}_{x(t)})_{t\in[0,1]}$ *lifted to a geometric* p*-rough path. However, for machine learning applications it is often beneficial to take* $d = \lfloor p \rfloor$ *to ensure convergence but to consider* $m > d$ *and find the optimal degree* m *from the data (e.g. by cross-validation).*

In analogy to the order $d = 1$ approximations discussed in Section 4, the central mathematical identity is now

$$\prod_{i=1}^{\ell-1} \sum_{m=0}^{d} \frac{(x(t_{i+1}) - x(t_i))^{\otimes m}}{m!} = \sum_{\boldsymbol{i} \sqsubseteq_d [\ell-1]} \frac{1}{\boldsymbol{i}!} \prod_{r=1}^{|\boldsymbol{i}|} (x(t_{i_r+1}) - x(t_i)) \approx \left(\int \mathrm{d}x^{\otimes m} \right)_{m \geq 0}$$

where we use Notation A.3.

Proposition B.7 *For* $x, y \in \mathcal{X}^+$, $d, m \geq 1$, $m \geq d$,

(a) $\mathrm{S}^+_{(d,m)}(\mathrm{k}_x) = 1 + \sum_{n=1}^{m} \sum_{\substack{\boldsymbol{i} \sqsubseteq_d [|x|-1] \\ |\boldsymbol{i}| = n}} \frac{1}{\boldsymbol{i}!} \prod_{r=1}^{|\boldsymbol{i}|} \nabla_{i} \mathrm{k}_x$,

(b) $\mathrm{k}^+_{(d,m)}(x, y) = 1 + \sum_{n=1}^{m} \sum_{\substack{\boldsymbol{i} \sqsubseteq_d [|x|-1] \\ \boldsymbol{j} \sqsubseteq_d [|y|-1] \\ |\boldsymbol{i}| = |\boldsymbol{j}| = n}} \frac{1}{\boldsymbol{i}!\boldsymbol{j}!} \prod_{r=1}^{|\boldsymbol{i}|} \nabla_{i_r, j_r} \mathrm{k}(x, y)$.

where $\nabla_i \mathrm{k}_x := \mathrm{k}_{x_{i+1}} - \mathrm{k}_{x_i} \in \mathcal{H}$.

Proof By definition

$$\mathrm{S}^+_{(d)}(\mathrm{k}_x) = \prod_{r=1}^{|x|-1} \sum_{n=0}^{d} \frac{1}{n!} (\nabla_r \mathrm{k}_x)^{\otimes n}.$$

Application of the (non-commutative) associative law yields

$$\prod_{r=1}^{|x|-1} \sum_{n=0}^{d} \frac{1}{n!} (\nabla_r \mathrm{k}_x)^{\otimes n} = \sum_{\boldsymbol{i} \sqsubseteq_d [|x|-1]} \frac{1}{\boldsymbol{i}!} \prod_{r=1}^{|\boldsymbol{i}|} \nabla_{i_r} \mathrm{k}_x \,.$$

Using the analogous expression for $\mathrm{S}^+_{(d)}(y)$, taking the scalar product while noting

$$\langle \prod_{r=1}^{|\boldsymbol{i}|} \nabla_{i_r} \mathrm{k}_x, \prod_{r'=1}^{|\boldsymbol{j}|} \nabla \mathrm{k}_{j_{r'}} \rangle = \delta_{|\boldsymbol{i}|,|\boldsymbol{j}|} \cdot \prod_{r=1}^{|\boldsymbol{i}|} \langle \nabla \mathrm{k}_{i_r}, \nabla_{j_r} \mathrm{k}_y \rangle.$$

and truncating at tensor degree m and restricting to $\boldsymbol{i} \sqsubseteq_d [|x| - 1]$ and $\boldsymbol{j} \sqsubseteq_d [|y| - 1]$ yields the claim. ∎

Remark B.8 *The appearance of the $\boldsymbol{i}!$ term together with $\boldsymbol{i} \sqsubseteq_d [|x|-1]$ makes a recursion formula more complex since one needs to keep track how many elements in $\boldsymbol{i}$ are equal in every recursion step. We give an effective algorithm in Section B.1 that relies on multi-way recursion.*

B.1. Computing the higher order discretized signature kernel

Algorithm 6 Evaluation of $\mathrm{k}^+_{(d,m)}$.
Input: Sequences $x, y \in \mathcal{X}^+$. A kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. A truncation degree m. An approximation order d with $1 \leq d \leq m$.
Output: $\mathrm{k}^+_{(d,m)}(x, y)$

Let $\ell_x \leftarrow |x| - 1$ and $\ell_y \leftarrow |y| - 1$
Compute an $(\ell_x \times \ell_y)$ array K such that $K[i, j] = \nabla_{i,j} \mathrm{k}(x, y)$
Initialize an $(m \times d \times d \times \ell_x \times \ell_y)$-array A, all entries zero
for $n = 2$ to m **do**
    $d' \leftarrow \min(d, n-1)$
    $A[n|1,1|:,:] \leftarrow K \cdot (1 + A[n-1|\Sigma, \Sigma| \boxplus +1, \boxplus + 1])$
    for $r = 2$ to d' **do**
        $A[n|r,1|:,:] \leftarrow \frac{1}{r} \cdot K \cdot A[n-1|r-1, \Sigma|:, \boxplus + 1]$
        $A[n|1,r|:,:] \leftarrow \frac{1}{r} \cdot K \cdot A[n-1|\Sigma, r-1| \boxplus +1, :]$
        for $s = 2$ to d' **do**
            $A[n|r,s|:,:] \leftarrow \frac{1}{rs} \cdot K \cdot A[n-1|r-1, s-1|:,:]$
        end for
    end for
end for
Compute $R \leftarrow 1 + A[m|\Sigma, \Sigma|\Sigma, \Sigma]$
Return R

Recall that multiplications of arrays in Algorithm 6 are entry-wise (not matrix multiplications). At the end of Algorithm 6, the array A contains as elements $A[m|r, s|i, j]$ the contributions from sub-sequences $\boldsymbol{i} \sqsubseteq [\ell_x], \boldsymbol{j} \sqsubseteq [\ell_y]$, beginning at i and j, with start-sequences $iii\ldots$ of length r and $jjj\ldots$ of length s, and of total length at most m.

Proposition B.9 *Let $x, y \in \mathcal{X}^+$, $m \geq 1$ and d with $1 \leq d \leq m$. Define $\ell_x = |x| - 1$, $\ell_y = |y| - 1$ and*

$$A^{n,r,s}_{i,j} := \sum_{\substack{\boldsymbol{i} \sqsubseteq_d [\ell_x] \\ \boldsymbol{j} \sqsubseteq_d [\ell_y] \\ |\boldsymbol{i}| = |\boldsymbol{j}| = n}} \frac{1}{\boldsymbol{i}!\boldsymbol{j}!} \prod_{\kappa=1}^{n} \nabla_{i_\kappa, j_\kappa} \mathrm{k}(x_{i_\kappa}, y_{j_\kappa})$$

for every $r, s \geq 1$ and $i \in [\ell_x]$, $j \in [\ell_y]$ where the sum over $\boldsymbol{i} = (i_1, \ldots, i_n), \boldsymbol{j} = (j_1, \ldots, j_n)$ is additionally restricted in the following way:

$$i = i_1 = i_2 = \cdots = i_r \neq i_{r+1} \quad \text{and} \quad j = j_1 = j_2 = \cdots = j_s \neq j_{s+1}.$$

By convention, set $A_{i,j}^{n,r,s} = 0$ *if* $r < n$ *or* $s < n$*. Then,*

$$\mathrm{k}^{+}_{(d,m)}(x,y) = 1 + \sum_{n=1}^{m}\sum_{i=1}^{\ell_x}\sum_{j=1}^{\ell_y}\sum_{r=1}^{d}\sum_{s=1}^{d} A_{i,j}^{n,r,s} \tag{5}$$

and the following equalities hold

$$A_{i,j}^{n,r,s} = \frac{1}{rs}\cdot\nabla_{i,j}\,\mathrm{k}(x,y)\cdot A_{i,j}^{n-1,r-1,s-1}, \tag{6}$$

$$A_{i,j}^{n,1,1} = \nabla_{i,j}\,\mathrm{k}(x,y)\cdot\left(1 + \sum_{i'>i}\sum_{j'>j}\sum_{r=1}^{d}\sum_{s=1}^{d} A_{i',j'}^{n-1,r,s}\right), \tag{7}$$

$$A_{i,j}^{n,r,1} = \frac{1}{r}\cdot\nabla_{i,j}\,\mathrm{k}(x,y)\cdot\sum_{j'>j}\sum_{s=1}^{d} A_{i,j'}^{n-1,r-1,s}, \tag{8}$$

$$A_{i,j}^{n,1,s} = \frac{1}{s}\cdot\nabla_{i,j}\,\mathrm{k}(x,y)\cdot\sum_{i'>i}\sum_{r=1}^{d} A_{i',j}^{n-1,r,s-1}, \tag{9}$$

for $r,s \geq 2$*,* $i \in [\ell_x]$*,* $j \in [\ell_y]$*.*

Proof By Proposition B.7

$$\begin{aligned}\mathrm{k}^{+}_{(d,m)}(x,y) &= 1 + \sum_{n=1}^{m}\sum_{\substack{\boldsymbol{i}\sqsubseteq_d[\ell_x]\\ \boldsymbol{j}\sqsubseteq_d[\ell_y]\\ |\boldsymbol{i}|=|\boldsymbol{j}|=n}} \frac{1}{\boldsymbol{i}!\boldsymbol{j}!}\prod_{r=1}^{n}\nabla_{i_r,j_r}\,\mathrm{k}(x,y)\\ &= 1 + \sum_{\substack{\boldsymbol{i}\sqsubseteq_d[\ell_x]\\ \boldsymbol{j}\sqsubseteq_d[\ell_y]}}\frac{1}{\boldsymbol{i}!\boldsymbol{j}!}\nabla_{i_1,j_1}\,\mathrm{k}(x,y)\,(1+\nabla_{i_2,j_2}\,\mathrm{k}(x,y)\,(1+\cdots(1+\nabla_{i_m,j_m}\,\mathrm{k}(x,y)))\cdots)\end{aligned}$$

and the first equality shows the identity (5) by explicitly summing over that starting point of sequences and the number of repeated elements. Equality (6) follows directly by definition of $A_{i,j}^{n,r,s}$ and that for $\boldsymbol{i} = (i_1, i_2, \ldots, i_n)$ with $d(\boldsymbol{i}) = r$ and $\boldsymbol{i}' = (i_2, \ldots, i_n)$ we have $\boldsymbol{i}! = \frac{1}{r}(\boldsymbol{i}'!)$. Equality (7) follows since

$$\begin{aligned}\{\boldsymbol{i} = (i_1,\ldots,i_n) : d(\boldsymbol{i}) = 0,\, i_1 = i\} &= \{(i,\boldsymbol{i}') : \boldsymbol{i}' = (i_1',\ldots,i_{n-1}'),\, i_1' > i\}\\ &= \bigcup_{r=0}^{d}\{(i,\boldsymbol{i}') : \boldsymbol{i}' = (i_1',\ldots,i_{n-1}'),\, i_1' > i,\, d(\boldsymbol{i}') = r\}\end{aligned}$$

where in above equalities we implicitly additionally assume that all indices $\boldsymbol{i}$, $\boldsymbol{i}'$ are increasing (that is, $i_1 \leq \cdots \leq i_n$, $i_1' \leq \cdots \leq i_{n-1}'$). Equalities (8),(9) follow similarly. ■

Remark B.10 *The computational cost of Algorithm 6 is* $O(d^2 m|x||y|)$ *elementary arithmetic operations (= the number of loop elements) and* $O(d^2|x||y|)$ *units of elementary storage (when freeing up space for array entries directly after the last time they are read out).*

Remark B.11 *The higher order Algorithm 6 can be combined with the low-rank techniques analogous to the treatment of the order $d = 1$ algorithm in Section 6 by applying the low-rank representation to the matrices/2D-arrays $A[m|i,j|:,:]$. Since all assignments and manipulations can be re-phrased in those matrices, the same strategy applies.*

References

Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. Online handwriting recognition with support vector machines-a kernel approach. In *Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on*, pages 49–54. IEEE, 2002.

Dan Crisan, Joscha Diehl, Peter Friz, and Harald Oberhauser. Robust filtering: multdimensional noise and multidimensional observation. *Annals of Applied Probability*, 23(5): 2139–2160, 2013.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.

Marco Cuturi. Fast global alignment kernels. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 929–936, 2011.

Marco Cuturi, J-P Vert, Oystein Birkenes, and Takashi Matsui. A kernel for time series based on global alignments. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–413. IEEE, 2007.

Joscha Diehl. Rotation invariants of two dimensional curves based on iterated integrals. *CoRR*, abs/1305.6883, 2013. URL `http://arxiv.org/abs/1305.6883`.

Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of statistical Software*, 31(7):1–24, 2009.

Ben Hambly and Terry J Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, 171(1):109–167, 2010.

David Haussler. Convolution kernels on discrete structures. Technical report, Citeseer, 1999.

Joseph B Kruskal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237, 1983.

Christina Leslie and Rui Kuang. Fast string kernels using inexact matching for protein sequences. *Journal of Machine Learning Research*, 2004.

Daniel Levin, Terry J Lyons, Hao Ni, et al. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260*, 2013.

Huma Lohdi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2002.

Terry J Lyons. *Differential equations driven by rough paths*. Springer Berlin Heidelberg New York, 2004.

Terry J Lyons, Hao Ni, and Harald Oberhauser. A feature set for streams and an application to high-frequency financial tick data. In *Proceedings of the 2014 International Conference on Big Data Science and Computing*, page 5. ACM, 2014.

Hiroshi Shimodaira Ken-ichi Noma. Dynamic time-alignment kernel in support vector machine. *Advances in neural information processing systems*, 14:921, 2002.

Anastasia Papavasiliou, Christophe Ladroue, et al. Parameter estimation for rough differential equations. *The Annals of Statistics*, 39(4):2047–2073, 2011.

Hiroaki Sakoe. Two-level dp-matching–a dynamic programming-based pattern matching algorithm for connected word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(6):588–595, 1979.

Hiroaki Sakoe and Seibi Chiba. A similarity evaluation of speech patterns by dynamic programming. In *Nat. Meeting of Institute of Electronic Communications Engineers of Japan*, page 136, 1970.

Christos Sapsanis, George Georgoulas, and Anthony Tzes. EMG based classification of basic hand movements based on time-frequency features. In *Control & Automation (MED), 2013 21st Mediterranean Conference on*, pages 716–722. IEEE, 2013.

Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

John Shawe-Taylor and Nello Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

Kilho Shin and Tetsuji Kuboyama. A generalization of Haussler's convolution kernel: mapping kernel. In *Proceedings of the 25th international conference on Machine learning*, pages 944–951. ACM, 2008.

Weixin Yang, Lianwen Jin, and Manfei Liu. Character-level chinese writer identification using path signature feature, dropstroke and deep CNN. abs/1505.04922, 2015. URL `http://arxiv.org/abs/1505.04922`.