Kernels for sequentially ordered data

Franz J. Király [*] [1] and Harald Oberhauser [†] [2]

[1] Department of Statistical Science, University College London,
Gower Street, London WC1E 6BT, United Kingdom
[2] Mathematical Institute, University of Oxford,
Andrew Wiles Building, Oxford OX2 6GG, United Kingdom

Abstract

We present a novel framework for kernel learning with sequential data of any kind, such as time series, sequences of graphs, or strings. Our approach is based on signature features which can be seen as an ordered variant of sample (cross-)moments; it allows to obtain a "sequentialized" version of any static kernel. The sequential kernels are efficiently computable for discrete sequences and are shown to approximate a continuous moment form in a sampling sense.

A number of known kernels for sequences arise as "sequentializations" of suitable static kernels: string kernels may be obtained as a special case, and alignment kernels are closely related up to a modification that resolves their open non-definiteness issue. Our experiments indicate that our signature-based sequential kernel framework may be a promising approach to learning with sequential data, such as time series, that allows to avoid extensive manual pre-processing.

[*] f.kiraly@ucl.ac.uk
[†] oberhauser@maths.ox.ac.uk

Contents

1. Introduction

Sequentially ordered data are ubiquitous in modern science, occurring as time series, location series, or, more generally, sequentially observed samples of numbers, vectors, and structured objects. They occur frequently in structured machine learning tasks, in supervised classification and regression as well as in forecasting, as well as in unsupervised learning.

Three stylized facts make learning with sequential data an ongoing challenge:

(A) Sequential data is usually very diverse, with wildly different features being useful. In the state-of-the-art, this is usually addressed by **manual extraction of hand-crafted features**, the combination of which is often very specific to the application at hand and does not transfer easily.

(B) Sequential data often occurs as **sequences of structured objects**, such as letters in text, images in video, graphs in network evolution, or heterogenous combination of all mentioned say in database or internet applications. This is usually solved by ad-hoc approaches adapted to the specific structure.

(C) Sequential data is often large, with sequences easily obtaining the length of hundreds, thousands, millions. Especially when there is one or more sequences per data point, the data sets quickly become **very huge**, and with them computational time.

In this paper, we present a novel approach to learning with sequential data based on a **joining of the theory of signatures/rough paths**, and **kernels/Gaussian processes**, addressing the points above:

(A) The **signature** of a path is a (large) collection of **canonical features** that can be intuitively described an ordered version of sample moments. They completely describe a sequence of vectors (provably), and make sequences of different size and length comparable. The use of signature features is therefore a straightforward way of avoiding manual feature extraction (Section 3).

(B) Combining signatures with the **kernel trick**, by considering the signature map as a feature map yields a kernel for sequences. It also allows **learning with sequences of structured objects** for which non-sequential kernels exist — consequently we call the process of obtaining a sequence kernel from a kernel for structured objects "kernel sequentialization" (Section 4).

(C) The **sequentialized kernel** can be **computed efficiently** via dynamic programming ideas similar to those known for string kernels (Section 5). The kernel formalism makes the computations further amenable to low-rank type speed-ups in kernel and Gaussian process learning such as Nyström-type and Cholesky approximations or inducing point methods (Section 8).

To sum up, we provide a canonical construction to transform any kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} :\to \mathbb{R}$ into a version for sequences $\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$, where we have denoted by $\mathcal{X}^+$ the set of arbitrary length sequences in $\mathcal{X}$. We call k^+ the **sequentialization of** k. This sequentialization is canonical in the sense that it converges to an inner product of ordered moments, the signature, when sequences in $\mathcal{X}^+$ converge to functions $[[0,1] \to \mathcal{X}]$ in a meaningful way. We will see that existing kernels for sequences such as string or alignment kernels are closely related to this construction.

We explore the practical use of the sequential kernel in experiments which show that sequentialization of non-linear kernels may be beneficial, and that the sequential kernel we propose can beat the state-of-the-art in sequence classification while avoiding extensive pre-processing. Below we give an informal overview of the main ideas, and a summary of related prior art.

1.1. Signature features, and their universality for sequences

Signatures are universal features for sequences, characterizing sequential structure by quantifying dependencies in their change, similar to sample moments. We showcase how to obtain such signature features for the simple example of a two-dimensional, smooth series

$$x : [0,1] \to \mathbb{R}^2, t \mapsto (a(t), b(t))^\top,$$

whose argument we interpret as "time". As with sample moments or sample cumulants of different degree, there are signature features of different degree, first degree, second degree, and so on. The first degree part of the signature is the average change in the series, that is,

$$\mathrm{S}_1(x) := \mathbb{E}_t[\dot{x}(t)] = \mathbb{E}_t\left[\begin{pmatrix}\dot{a}(t)\\ \dot{b}(t)\end{pmatrix}\right] = \begin{pmatrix}\int_0^1 \mathrm{d}a(t)\\ \int_0^1 \mathrm{d}b(t)\end{pmatrix} = \begin{pmatrix}a(1)-a(0)\\ b(1)-b(0)\end{pmatrix},$$

where we have written $\mathrm{d}a(t) := \dot{a}(t)\mathrm{d}t, \mathrm{d}b(t) := \dot{b}(t)\mathrm{d}t$. The second degree part of the signature is the (non-centered) covariance of changes at two subsequent time points, that is, the expectation

$$\begin{aligned}\mathrm{S}_2(x) &:= \frac{1}{2}\mathbb{E}_{t_1<t_2}[\dot{x}(t_1)\cdot\dot{x}(t_2)^\top] = \frac{1}{2}\mathbb{E}_{t_1<t_2}\left[\begin{pmatrix}\dot{a}(t_1)\dot{a}(t_2) & \dot{a}(t_1)\dot{b}(t_2)\\ \dot{b}(t_1)\dot{a}(t_2) & \dot{b}(t_1)\dot{b}(t_2)\end{pmatrix}\right]\\ &= \begin{pmatrix}\int_0^1\int_0^{t_1}\mathrm{d}a(t_2)\mathrm{d}a(t_1) & \int_0^1\int_0^{t_1}\mathrm{d}a(t_2)\mathrm{d}b(t_1)\\ \int_0^1\int_0^{t_1}\mathrm{d}b(t_2)\mathrm{d}a(t_1) & \int_0^1\int_0^{t_1}\mathrm{d}b(t_2)\mathrm{d}b(t_1)\end{pmatrix},\end{aligned}$$

where the expectations in the first line are uniformly over time points in chronological order $t_1 \leq t_2$ (that is, t_1, t_2 is the order statistic of two points sampled from the uniform distribution on $[0,1]$). This is equivalent to integration over the so-called 2-order-simplex $\{t_1, t_2 : 0 \leq t_1 \leq t_2 \leq 1\}$ in the second line, up to a factor of $1/2$ corresponding to the uniform density (we put it in front of the expectation and not its inverse in front of the integral to obtain an exponential generating function later on).

Note that the second order signature is different from the second order moment matrix of the infinitesimal changes by the chronological order imposed in the expectation. Similarly, one defines the degree M part of the signature as the M-th order moment tensor of the infinitesimal changes, where expectation is taken over chronologically ordered time points (which is a tensor of degree M). A basis-free definition over an arbitrary RKHS is given in Section 3.2. Note that the signature tensors are not symmetric, similarly to the second order matrix in which the number arising from the $\dot{b}(t_1)\dot{a}(t_2)$ term is in general different from the number obtained from the $\dot{a}(t_1)\dot{b}(t_2)$ term.

The signature features are in close mathematical analogy to moments and thus polynomials on the domain of multi-dimensional series. Namely, one can show:

- A (sufficiently regular) series is (almost) uniquely determined by their signature - this is not true for higher order moments or cumulants without the order structure (recall that these almost uniquely determine the distribution of values, without order structure).

- Any (sufficiently regular real-valued) function f on series can be arbitrarily approximated by a function linear in signature features, that is for a non-linear functionals f of our two-dimensional path $x = (a, b)^\top$,

  $$f(x) \approx \alpha + \sum \beta_{i_1,\ldots,i_M} \int \mathrm{d}x_{i_1}\cdots\mathrm{d}x_{i_M},$$

  where the sum runs over M and $(i_1, \ldots, i_M) \in \{1,2\}^M$ and we denote $x_1 := a, x_2 := b$. Note that x is the indeterminate here, that $\int \mathrm{d}x_{i_1}\cdots\mathrm{d}x_{i_M}$ is the degree M part of the signature and approximation is over a compact set of different paths x. The exact statements are given in Section 3.

From a methodological viewpoint, these assertions mean that not only are signature features rich enough to capture all relevant features of the sequential data, but also that any practically relevant feature can be expressed *linearly* in the signature, addressing point (A) in the sense of a universal methodological approach.

Unfortunately, native signature features, in the form above are only practical in low dimension and low degree M: already in the example above of a two-dimensional path x, there are 2^M (scalar) signature features of degree M, in general computation of a larger number of signature features is infeasible, point (C) Further, all data are discrete sequences not continuous, and possibly of objects which are not necessarily real vectors; point (B).

1.2. The sequential kernel and sequentialization

The two issues mentioned can be addressed by the kernel trick — more precisely, by the kernel trick applied twice: once, to cope with the combinatorial explosion of signature features, akin to the polynomial kernel which prevents computation of an exponential number of polynomial features; a second time, to allow treatment of sequences of arbitrary objects. This double kernelization addresses point (B), and also the combinatorial explosion of the feature space. An additional discretization-approximation, which we discuss in the next paragraph below, makes the so-defined kernel amenable to efficient computation.

We describe the two kernelization steps. The first kernelization step addresses the combinatorial explosion. It simply consists of taking the scalar product of signature features as kernel, and then observing that this scalar product of integrals is an integral of scalar products. More precisely, this kernel, called *signature kernel* can be defined as follows, continuing the example with two two-dimensional sequences $t \mapsto x(t) = (a(t), b(t))^\top$, $t \mapsto \bar{x}(t) = (\bar{a}(t), \bar{b}(t))^\top$ as above:

$$\mathrm{K}^\oplus(x, \bar{x}) := \langle \mathrm{S}(x), \mathrm{S}(\bar{x})\rangle = 1 + \langle \mathrm{S}_1(x), \mathrm{S}_1(\bar{x})\rangle + \langle \mathrm{S}_2(x), \mathrm{S}_2(\bar{x})\rangle + \cdots .$$

The scalar product of S_1-s (vectors in $\mathbb{R}^2$) is the Euclidean scalar product in $\mathbb{R}^2$, the scalar product of S_2-s (matrices in $\mathbb{R}^{2\times 2}$) is the trace product in $\mathbb{R}^2$, and so on (with higher degree tensor trace products). The "1" is an "S_0"-contribution (for mathematical reasons becoming apparent in paragraph 1.3 below).

For the first degree contribution to the signature kernel, one now notes that

$$\begin{aligned}\langle \mathrm{S}_1(x), \mathrm{S}_1(\overline{x})\rangle &= \mathbb{E}_s\left[\dot{a}(s)\right] \cdot \mathbb{E}_t\left[\dot{\overline{a}}(t)\right] + \mathbb{E}_s\left[\dot{b}(s)\right] \cdot \mathbb{E}_t\left[\dot{\overline{b}}(t)\right] \\ &= \mathbb{E}_{s,t}\left[\dot{a}(s) \cdot \dot{\overline{a}}(t) + \dot{b}(s) \cdot \dot{\overline{a}}(t)\right] \\ &= \mathbb{E}_{s,t}\left[\langle \dot{x}(s), \dot{\overline{x}}(t)\rangle\right].\end{aligned}$$

In analogy, one computes that the second degree contribution to the signature kernel evaluates to

$$\langle \mathrm{S}_2(x), \mathrm{S}_2(\overline{x})\rangle = \frac{1}{2!^2}\mathbb{E}_{s_1<s_2, t_1<t_2}\left[\langle \dot{x}(s_1), \dot{\overline{x}}(t_1)\rangle \cdot \langle \dot{x}(s_2), \dot{\overline{x}}(t_2)\rangle\right].$$

Similarly, for a higher degree M, one obtains a product of M scalar products in the expectation.

The presentation is not only reminiscent of the polynomial kernel in how it copes with the combinatorial explosion, it also directly suggests the second kernelization to cope with sequences of arbitrary objects: since the sequential kernel is now entirely expressed in scalar products in $\mathbb{R}^2$, the scalar products in the expectation may be replaced by any kernel $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, of arbitrary objects, yielding a sequential kernel, now for sequences in $\mathcal{X}$, given as

$$\mathrm{k}^\oplus(x, \bar{x}) := 1 + \frac{1}{1!^2}\mathbb{E}_{s,t}\left[\mathrm{k}(\dot{x}(s), \dot{\overline{x}}(t))\right] + \frac{1}{2!^2}\mathbb{E}_{s_1<s_2, t_1<t_2}\left[\mathrm{k}(\dot{x}(s_1), \dot{\overline{x}}(t_1)) \cdot \mathrm{k}(\dot{x}(s_2), \dot{\overline{x}}(t_2))\right] + \frac{1}{3!^2}\mathbb{E}\ldots \tag{1.1}$$

(for expository convenience we assume here that differentials in $\mathcal{X}$ are defined which in general is untrue, see Section 4.3 for the general statement). Note that (1.1) can be seen as a process that takes any kernel k on $\mathcal{X}$, and makes it into a kernel $\mathrm{k}^\oplus$ on $\mathcal{X}$-sequences, therefore we term it "sequentialization" of the kernel k. This addresses point (B), and can be found in more detail in Section 4.

1.3. Efficient computation and discretization

An efficient way of evaluating the sequential kernel is suggested by a second observation, closely related to (and generalizing) Horner's method of evaluating polynomials. Note that the sequential kernel can be written as an iterated conditional expectation

$$\mathrm{k}^\oplus(x, \bar{x}) = \left(1 + \mathbb{E}_{s_1,t_1}\left[\mathrm{k}(\dot{x}(s_1), \dot{\overline{x}}(t_1)) \cdot \left(1 + \frac{1}{2^2}\mathbb{E}_{s_1<s_2, t_1<t_2}\left[\mathrm{k}(\dot{x}(s_2), \dot{\overline{x}}(t_2)) \cdot \left(1 + \frac{1}{3^2}\mathbb{E}_{s_2<s_3, t_2<t_3}[\ldots]\right)\right]\right)\right]\right).$$

The iterated expectation directly suggests a discretization by replacing expectations by sums, such as

$$\left(1+\frac{1}{n^2}\sum_{s_1,t_1}\left[\mathrm{k}(\dot{x}(s_1),\dot{\overline{x}}(t_1))\cdot\left(1+\frac{1}{(2n)^2}\sum_{s_1<s_2,t_1<t_2}\left[\mathrm{k}(\dot{x}(s_2),\dot{\overline{x}}(t_2))\cdot\left(1+\frac{1}{(3n)^2}\sum_{s_2<s_3,t_2<t_3}[\dots]\right)\right]\right)\right]\right),$$

where the sums range over a discrete set of points $s_i, t_i \in [0,1]$, $i = 1,\dots L$. A reader familiar with string kernels will immediately notice the similarity: the sequential kernel can in fact be seen as infinitesimal limit of a string kernel, and the (vanilla) string kernel can be obtained as a special case (see Section 6). As a final subtlety, we note that the derivatives of $x, \overline{x}$ will not be known in observations, therefore one needs to replace $\mathrm{k}(\dot{x}(s_i),\dot{\overline{x}}(t_j))$ by a discrete difference approximation

$$\mathrm{k}(x(s_{i+1}),\overline{x}(t_{j+1}))+\mathrm{k}(x(s_i),\overline{x}(t_j))-\mathrm{k}(x(s_i),\overline{x}(t_{j+1}))-\mathrm{k}(x(s_{i+1}),\overline{x}(t_j))$$

where s_i, s_{i+1} resp. t_j, t_{j+1} denote adjacent support values of the discretization.

Our theoretical results in Section 5 show that the discretization, as described above, converges to the continuous kernel, with a convergence order linear in the sampling density. Moreover, similarly to the Horner scheme for polynomials (or fast string kernel techniques), the iterated sum-product can be efficiently evaluated by dynamical programming techniques on arrays of dimension three, as outlined in Section 8. The computational complexity is quadratic in the length of the sequences and linear in the degree of approximation, and can be further reduced to linear complexity in both with low-rank techniques.

This addresses the remaining point (C) and therefore yields an efficently computable, canonical and universal kernel for sequences of arbitrary objects.

1.4. Prior art

Prior art relevant to learning with sequential data may be found in three areas:

1. dynamic programming algorithms for sequence comparison in the engineering community,
2. kernel learning and Gaussian processes in the machine learning community
3. rough paths in the stochastic analysis community.

The dynamic programming literature (1) from the 70's and 80's has inspired some of the progress in kernels (2) for sequential data over the last decade, but to our knowledge so far no connections have been made between these two, and (3), even though (3) pre-dates kernel literature for sequences by more than a decade. Beyond the above, we are not aware of literature in statistics of time series 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 (this is equally true for forecasting methods). Though, simple as such reduction to more classic situations may be, it constitutes an important baseline, since only in comparison one can infer that the ordering was informative or not.

Dynamic programming for sequence comparison. The earliest occurrence in which the genuine order information in sequences is used for learning can probably be found in the work of Sakoe et al [29, 30] which introduces the idea of using editing or distortion distances to compare sequences of different length, and to efficiently determine such distances via dynamic programming strategies. 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 [9, 16].

Though it may need to be said that sequence comparison methods in their pure form — namely an efficiently computable distance between sequences — have remained somewhat restricted in that they can only be directly adapted only to relatively heuristic distance-based learning algorithms, by definition. This may be one of the reasons why sequence comparison/dynamic time warping methods have not given rise to a closed learning theory, and why in their original practical application, speech recognition and speech classification, they have later been superseded by Hidden Markov Models [26] and more recently by neural network/deep learning methodologies [15] as gold standard.

A possible solution of the above-mentioned shortcomings has been demonstrated in kernel learning literature.

Kernels for sequences. Kernel learning is a relatively new field, providing a general framework to make non-linear data of arbitrary kind amenable to classical and scalable linear algorithms such as regression or the support vector machine in a unified way, by using a non-linear scalar product: this strategy is called "kernelization"; see [32] or [33]. Mathematically, there are close relations to Gaussian process theory [27] which is often considered as a complimentary viewpoint to kernels, and aspects of spatial geostatistics [4], particularly Kriging, an interpolation/prediction method from the 60's [23] which has been re-discovered 30 years later in the form of Gaussian process regression [35]. In all three incarnations, coping with a specific kind of data practically reduces to finding a suitable kernel (= covariance function), or a family of kernels for the type of data at hand — after which one can apply a ready arsenal of learning theory and non-linear methodology to such data. In this sense, providing suitable kernels for sequences has proved to be one of the main strategies in removing the shortcomings of the sequence comparison approach.

Kernels for strings, that is, sequences of symbols, were among the first to be considered [14]. Fast dynamic programming algorithms to compute string kernels were obtained a few years later [17, 19]. Almost in parallel and somewhat separately, kernels based on the above-mentioned dynamic time warping approach were developed, for sequences of arbitrary objects [1, 24]. A re-formulation/modification led to the so-called global alignment kernels [6], for which later fast dynamic programming algorithms were found [5] as well. An interesting subtle issue common to both strains was that initially, the dynamic programming algorithms found were quadratic in the length of the sequences, and only later linear complexity algorithms were devised: for string kernels, the transition was made in [17], while for the sequence matching strain, this became only possible after passing to the global alignment kernels [5].

Looking from a general perspective: while in hindsight all of the mentioned kernels can be viewed from Haussler's original, visionary relation-convolution kernel framework, and all above-mentioned kernels for sequences, in some form, admit fast dynamic programming algorithms, existing literature provides no unifying view on kernels for sequences: the exact 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, suffer from the fact that they are not proper kernels, failing to be positive definite.

In this paper, we attempt to resolve these issues. More precisely, the string kernel will arise as a special case, and the global alignment kernel as a deficient version of our new signature kernel, built on the theory of signatures and rough paths from stochastic analysis.

Iterated integrals, signatures and rough paths. Series of iterated integrals are a classic mathematical object that plays a fundamental role in many areas like control theory, combinatorics, homotopy theory, Feynman–Dyson–Schwinger theory in physics and more recently probability theory. We refer to [20, Section "Historic papers", p97] for a bibliography of influential articles. This series, or certain aspects of it, are treated under various names in the literature like "Magnus expansion", "time ordered exponential", or the one we chose which comes from Lyons' rough path theory: "the signature of a path". The reason we work in the rough path setting is that it provides a concise mathematical framework that clearly separates analytic and algebraic aspects, applies in infinite dimensional spaces like our RKHS and is robust under noise; we refer to [8, 20, 22] as introductions. The role of the signature as a "non-commutative exponential" plays a guiding principle for many recent developments in stochastic analysis, though it might be less known outside this community.

The major application of rough path theory was and still is to provide a robust understanding of differential equations that are perturbed by noise, going beyond classic Ito-calculus; Hairer's work on regularity structures [12] which was recently awared a Fields medal can be seen as vast generalization of such ideas. The interpretation of the signature in a statistical context is more recent: work of Papavasiliou and Ladroue [25] applies it to SDE parameter estimation; work of Gyurko, Hao, Lyons, Oberhauser [11, 18, 21] applies it to forecasting and classifcation of financial time series using linear and logistic regression; work of Diehl [7] and Graham [10] uses signature features for handwritten digit recognition, see also [36] for more recent state of the art results.

The interpretation of the signature as an expectation already occurs as a technical Lemma 3.9 in [13]. A scalar product formula for the norm, somewhat reminiscent of that of the sequential kernel, can be found in the same Lemma. Similarly we would like to mention the Computational Rough Paths package [3], that contains C++ code to compute signature features directly for $\mathbb{R}^d$-valued paths. However, it does not contain specialized code to calculate inner products of signature features directly. The Horner-type algorithm we describe in Section 1.3 gives already significant speed improvements when it is applied to paths in finite dimensional, linear spaces (that is, the sequentialization of the Euclidean inner product $\mathrm{k}(\cdot,\cdot)=\langle\cdot,\cdot\rangle_{\mathbb{R}^d}$; see Remark 8.1).

Remark 1.1. *The above overview contains a substantial number of examples in which independent or parallel development of ideas related to sequential data has occurred, possibly due to researchers being unaware of similar ideas in communities socially far from their own. We are aware that this could therefore also be the case for this very paper, unintended. We are thus especially thankful for any pointers from readers of this pre-print version that help us give credit where credit is due.*

Acknowledgement

HO is grateful for support by the Oxford-Man Institute of Quantitative finance.

2. Notation for ordered data

We introduce recurring notation.

Definition 2.1. *The set of natural numbers, including* 0*, will be denoted by* $\mathbb{N}$.

Definition 2.2. *Let A be a set and $L \in \mathbb{N}$. We denote*

1. *the set of integers between* 1 *and L by* $[L] := \{1,2,\ldots,L\}$,
2. *the set of ordered L-tuples in A as usual by* $A^L := \{\boldsymbol{a} = (a_1,\ldots,a_L) : a_1,\ldots,a_L \in A\}$,
3. *the set of such tuples, of arbitrary but finite length, by* $A^+ := \bigcup_{L\in\mathbb{N}} A^L$ *where by convention* $A^0 = \emptyset$.

Moreover, we use the following index notation for $\boldsymbol{a} = (a_1,\ldots,a_L) \in A^+$,

1. $\ell(\boldsymbol{a}) := L$,
2. $\#\boldsymbol{a}$ *the count of the most frequent item in* $\boldsymbol{a}$,
3. $\boldsymbol{a}[i] := a_i$ *for* $i = 1,\ldots,L$,
4. $\boldsymbol{a}[1:N] := (\boldsymbol{a}[1],\ldots,\boldsymbol{a}[N])$ *for* $N \in [L]$,
5. $\boldsymbol{a}[\boldsymbol{i}] := (\boldsymbol{a}[i_1],\ldots,\boldsymbol{a}[i_N]) \in A^N$ *for* $\boldsymbol{i} = (i_1,\ldots,i_N) \in [L]^N$,
6. $\boldsymbol{a}! := n_1!\cdots n_k!$ *if* $\boldsymbol{a}$ *consists of* $k = |\{a_1,\ldots,a_L\}|$ *different elements in A and* $n_1,\ldots,n_k$ *denote the number of times they occur in* $\boldsymbol{a}$,

7. $f(\boldsymbol{a}) = \big(f(a_1),\ldots,f(a_L)\big) \in B^L$ *for* $f \in [A \to B]$.

In the case that $A \subset \mathbb{R}$, we can define subsets of A^+ that consist of increasing tuples. These tuples play an important role for calculations with the signature features.

Definition 2.3. *Let $A \subset \mathbb{R}$ and $M \in \mathbb{N}$. We denote the set of monotonously ordered M-tuples in S by*

$$\Delta^M(A) := \{\boldsymbol{u} \in A^M : \boldsymbol{u}[1] \leq \boldsymbol{u}[2] \leq \cdots \leq \boldsymbol{u}[M]\}.$$

We denote the union of such tuples by $\Delta(A) := \bigcup_{M\in\mathbb{N}} \Delta^M(A)$ where again by convention $\Delta^0(A) = \emptyset$. We call $\Delta^M(A)$ the order M simplex on A, and $\Delta(A)$ the order simplex on A. A monotonously ordered tuple $\boldsymbol{u} \in \Delta^M(A)$ is called strict if $\boldsymbol{u}[i] \lneq \boldsymbol{u}[i+1]$ for all $i \in [M-1]$. The index notation of Definition 2.2 applies also to $\boldsymbol{u} \in \Delta(A)$, understanding that $\Delta(A) \subseteq A^+$ if $A \subseteq \mathbb{R}$.

Remark 2.4. *Above definition of order simplex is slightly different from the usual one, in which one takes $A = [0,1]$, the counting starts at t_0, and one has $t_0 = 0$ and $t_M = 1$.*

The following notation is less standard, but becomes very useful for the algorithms and recursions that we discuss.

Definition 2.5. *Let $\boldsymbol{a}, \boldsymbol{b} \in A^+$, $D \in \mathbb{N}$. We use the following notation:*

- $\boldsymbol{a} \sqsubseteq \boldsymbol{b}$ *if there is a $\boldsymbol{i} \in \Delta(\mathbb{N})$ such that $\boldsymbol{a} = \boldsymbol{b}[\boldsymbol{i}]$,*
- $\boldsymbol{a} \sqsubset \boldsymbol{b}$ *if there is strictly ordered tuple $\boldsymbol{i} \in \Delta(\mathbb{N})$ such that $\boldsymbol{a} = \boldsymbol{b}[\boldsymbol{i}]$,*
- $\boldsymbol{a} \sqsubseteq_D \boldsymbol{b}$ *if there is a $\boldsymbol{i} \in \Delta(\mathbb{N})$ such that $\boldsymbol{a} = \boldsymbol{b}[\boldsymbol{i}]$ and $\#\boldsymbol{a} \leq D$.*

For $L \in \mathbb{N}$ we also the notation

1. $\boldsymbol{a} \sqsubseteq [L]$ *if* $\boldsymbol{a} \in \Delta([L])$*,*
2. $\boldsymbol{a} \sqsubset [L]$ *if $\boldsymbol{a} \in \Delta([L])$ and $\boldsymbol{a}$ is strict (that is $\boldsymbol{a}[i] \lneq \boldsymbol{a}[i+1]$ for all $i \in [\ell(\boldsymbol{a})-1]$).*

3. Signature features: ordered moments for sequential data

In this section, the signature features are introduced in a mathematically precise way, and the properties which make them canonical features for sequences are derived. We refer the interested reader to [20] for further properties of signatures and its use in stochastic analysis.

As outlined in introductory Section 1.1, the signature can be understood as an ordered and hence non-commutative variant of sample moments. If we model ordered data by a function $x \in [[0,1] \to \mathbb{R}^n]$, the signature features are obtained as iterated integrals of x and the M-times iterated integral can be interpreted as a M-th ordered moment; examples of such features for $M = 1,2$ are the (non-commutative) integral moments

$$\mathrm{S}_1 : [[0,1] \to \mathbb{R}^n] \to \mathbb{R}^n\,,\ x \mapsto \int_0^1 \mathrm{d}x(t), \quad \text{or} \quad \mathrm{S}_2 : [[0,1] \to \mathbb{R}^n] \to \mathbb{R}^{n\times n}\,,\ x \mapsto \int_0^1 \int_0^{t_2} \mathrm{d}x(t_1)\mathrm{d}x(t_2)^\top$$

(where the domain of x will be restricted so the integrals are well-defined). The more general idea of signature features, made mathematically precise, is to consider the integral moments

$$\mathrm{S}_M(x) = \left(\int_{\Delta^M} \mathrm{d}x_{i_1}(t_1)\cdots \mathrm{d}x_{i_M}(t_M)\right)_{i_1,\ldots,i_M\in\{1,\ldots,n\}} \in \mathbb{R}^{n\times\cdots\times n} \tag{3.1}$$

where the integration is over the M-simplex Δ^M, i.e., all ordered sequences $0 \leq t_1 \leq t_2 \leq \cdots \leq t_M \leq 1$, and the choice of index $(i_1,\ldots,i_M)$ parametrises the features. The features $\mathrm{S}_1(x), \mathrm{S}_2(x), \ldots$ are all element of a (graded) linear space and a kernel for sequential data may then be obtained from taking the scalar product over the signature features.

The section is devoted to a description and charactarization of these signature features and the kernel obtained from it. This is done in a basis-free form which allows treatment of ordered data $x : [0,1] \to \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space (over $\mathbb{R}$) not necessarily identical to $\mathbb{R}^n$. This will allow considering ordered versions of data, the non-ordered variant of which may be already encoded via its reproducing kernel Hilbert space $\mathcal{H}$.

For illustration, the reader is invited to keep the prototypical case $\mathcal{H} = \mathbb{R}^n$ in mind.

3.1. Paths of bounded variation

The main object modelling a (generative) datum will be a continuous sequence, a path $x : [0,1] \to \mathcal{H}$ where $\mathcal{H}$ is a Hilbert space which is fixed in the following. For reasons of regularity (integrals need to converge), we will restrict ourselves to paths $[[0,1] \to \mathcal{H}]$ which are continuous and of bounded length[1] (called "variation", as defined below).

The variation of a path is defined as the supremum of variations of discrete sub-sequences taken from the path:

Definition 3.1. *For an ordered tuple $\boldsymbol{x} \in \mathcal{H}^{L+1}$, we define:*

(i) $\nabla \boldsymbol{x} \in \mathcal{H}^L$ *as* $\nabla \boldsymbol{x}[i] := \boldsymbol{x}[i+1] - \boldsymbol{x}[i]$, $i \in [L]$,

(ii) $\operatorname{mesh}(\boldsymbol{x}) := \max_i \|(\nabla \boldsymbol{x})[i]\|$,

(iii) $\mathrm{V}(\boldsymbol{x}) := \sum_{i=1}^L \|\nabla \boldsymbol{x}\|_{\mathcal{H}}$.

We call $\nabla \boldsymbol{x}$ the first difference sequence of $\boldsymbol{x}$, we call $\operatorname{mesh}(\boldsymbol{x})$ *the mesh of $\boldsymbol{x}$, and* $\mathrm{V}(\boldsymbol{x})$ *the variation of $\boldsymbol{x}$.*

Definition 3.2. *We define the variation of $x \in [[0,1] \to \mathcal{H}]$ as*

$$\mathrm{V}(x) := \sup_{\boldsymbol{t} \in \Delta([0,1])} \mathrm{V}(x(\boldsymbol{t})).$$

The mapping x is said to be of bounded variation if $\mathrm{V}(x) < \infty$.

A bounded variation path, as the name says, is a path with bounded variation.

Definition 3.3. *Let $[a,b] \subseteq [0,1]$. We denote the set of $\mathcal{H}$-valued paths of bounded variation on $[a,b]$ by*

$$\mathrm{BV}([a,b],\mathcal{H}) := \{x \in C([a,b],\mathcal{H}) \,:\, \mathrm{V}(x) < \infty\}.$$

When $[a,b] = [0,1]$ we often omit the qualifier $[0,1]$ and write $\mathrm{BV}(\mathcal{H})$.

Definition 3.4. *Let E be a linear space and denote with $L(\mathcal{H},E)$ the set of continuous linear maps from $\mathcal{H}$ to E. Given $x \in \mathrm{BV}([0,1],\mathcal{H})$ and $y \in \mathrm{BV}([0,1], L(\mathcal{H}))$, the Riemann–Stieltjes integral of y over $[a,b] \subseteq [0,1]$ is defined as the element in E given as*

$$\int_a^b y \,\mathrm{d}x := \lim_{\substack{\boldsymbol{t} \in \Delta([a,b]) \\ \operatorname{mesh}(\boldsymbol{t}) \to 0}} \sum_{i=1}^{\ell(\boldsymbol{t})-1} y(\boldsymbol{t}[i])(\nabla x(\boldsymbol{t}[i])).$$

We also use the shorter notation $\int y \,\mathrm{d}x$ when the integration domain is clear from the context.

As in the finite-dimensional case, above is indeed well-defined, that is the Riemann sums converge and the limit is independent of the sequence $\boldsymbol{t}$; see [20, Theorem 1.16] for a proof of a more general result. Note that the integral itself is in general not a scalar, but an element of the Hilbert space $\mathcal{H}$.

[1] Considering bounded variation paths is slightly restrictive as it excludes samples from stochastic process models such as the Wiener process/Brownian motion. The theory may be extended to such sequences at the cost of an increase in technicality. For reading convenience, this will be done only at a later stage.

3.2. Signature integrals in Hilbert spaces

With the Riemann–Stieltjes integral, we can define the signature integrals in a basis-free way. For this, note that the Riemann–Stieltjes integral of a bounded variation path is again a bounded variation path.

Definition 3.5. *Let $[a,b] \subseteq [0,1]$ and $x \in \mathrm{BV}([a,b],\mathcal{H})$. We define:*

(i) $\int_{\Delta^1([a,b])} \mathrm{d}x^{\otimes 1} := \int_a^b \mathrm{d}x = x(b) - x(a)$,

(ii) $\int_{\Delta^M([a,b])} \mathrm{d}x^{\otimes M} := \int_a^b \left(\int_{\Delta^{M-1}([a,s])} \mathrm{d}x^{\otimes(M-1)}\right) \otimes \mathrm{d}x(s)$ *for integers* $M \geq 2$.

We call $\int_{\Delta^M([a,b])} \mathrm{d}x^{\otimes M}$ the M-th iterated integral of x on $[a,b]$. When $[a,b]=[0,1]$ we often omit the qualifier $[0,1]$ and write Δ^M for $\Delta^M([0,1])$.

In the prototypical case $\mathcal{H} = \mathbb{R}^n$, the first iterated integral is a vector in $\mathbb{R}^n$, the second iterated integral is a matrix in $\mathbb{R}^{n\times n}$, the third iterated integral is a tensor in $\mathbb{R}^{n\times n\times n}$ and so on. For the case of arbitrary Hilbert spaces, we need to introduce some additional notation to write down the space where the iterated integrals live:

Definition 3.6. *We denote by $\mathcal{H}\otimes\mathcal{H}$ the tensor (=outer product) product of $\mathcal{H}$ with itself. Instead of $\mathcal{H}\otimes\mathcal{H}\otimes\cdots\otimes\mathcal{H}$ (M times), we also write $\mathcal{H}^{\otimes M}$. By convention, $\mathcal{H}^{\otimes 0} = \mathbb{R}$.*

The M-th iterated integral of x is an element of $\mathcal{H}^{\otimes M}$, a tensor of degree M. The natural space of iterated integrals of all degrees together is the tensor power algebra the Hilbert space $\mathcal{H}$:

Definition 3.7. *The tensor power algebra over $\mathcal{H}$ is defined as $\bigoplus_{M=0}^{\infty}\mathcal{H}^{\otimes M}$, addition $+$, multiplication $*$ and an inner product $\langle\cdot,\cdot\rangle_{\mathrm{T}(\mathcal{H})}$ are defined by canonical extension from $\mathcal{H}$, that is:*

$$(g_0,g_1,\dots,g_M,\dots)+(h_0,h_1,\dots,h_M,\dots) := (g_0+h_0\; g_1+h_1,\dots\; g_M+h_M\;,\dots)\,.$$

$$(g_0,g_1,\dots,g_M,\dots)*(h_0,h_1,\dots,h_M,\dots) := \left(g_0h_0\,,\; g_0\otimes h_1+g_1\otimes h_0\,,\;\dots\,,\;\sum_{i=0}^{M} g_i\otimes h_{M-i}\,,\;\dots\right),$$

$$\langle(g_0,g_1,\dots,g_M,\dots),(h_0,h_1,\dots,h_M,\dots)\rangle_{\mathrm{T}(\mathcal{H})} := \langle g_0,h_0\rangle_{\mathbb{R}} + \langle g_1 h_1\rangle_{\mathcal{H}} + \cdots + \langle g_M,h_M\rangle_{\mathcal{H}^{\otimes M}} + \dots.$$

The elements in the tensor power algebra with with finite norm $\|g\|_{\mathrm{T}(\mathcal{H})} = \sqrt{\langle g,g\rangle_{\mathrm{T}(\mathcal{H})}} < \infty$ are denoted by $\mathrm{T}(\mathcal{H})$[2]. *Further, we canonically identify $\mathcal{H}^{\otimes M}$ with the sub-Hilbert space of $\mathrm{T}(\mathcal{H})$ that contains exactly elements of the type $g = (0,\dots,0,g_M,0,\dots)$, which we call homogenous (of degree k). Under this identification, for $g \in \mathrm{T}(\mathcal{H})$, we will also write, for reading convenience,*

$$g_0+g_1+g_2+\dots \quad \textit{instead of} \quad (g_0,g_1,g_2,\dots),$$

where we may opt to omit zeros in the sum. We adopt an analogue use of sum and product signs $\sum,\prod$.

Example 3.8. *We work out tensor algebra multiplication and scalar product in the case of the prototypical example $\mathcal{H}=\mathbb{R}^n$. Consider homogenous elements of degree 2: it holds that $\mathcal{H}^{\otimes 2} = \mathbb{R}^n\otimes\mathbb{R}^n \cong \mathbb{R}^{n\times n}$, and the tensor product of two vectors is $v\otimes w = vw^\top$. The trace product on $\mathbb{R}^{n\times n}$ is induced by the Euclidean scalar product, since $\langle v_1w_1^\top, v_2w_2^\top\rangle = \langle v_1,v_2\rangle\langle w_1,w_2\rangle$. Homogenous elements of degree 3 are similar: it holds that $\mathcal{H}^{\otimes 3}\cong\mathbb{R}^{n\times n\times n}$, and the tensor product of three vectors $u\otimes v\otimes w$ is a tensor T of degree* 3, *where $T_{ijk} = u_iv_jw_k$ for all $i,j,k\in[n]$. The scalar product on $\mathbb{R}^{n\times n\times n}$ is the tensor trace product, $\langle T,T'\rangle_{\mathcal{H}^{\otimes 3}} = \sum_{i,j,k=1}^n T_{ijk}T'_{ijk}$. An element $g\in\mathrm{T}(\mathcal{H})$ is of the form*

$$g = c+v+M+T+\dots,\quad \textit{where } c\in\mathbb{R}, v\in\mathbb{R}^n, M\in\mathbb{R}^{n\times n}, T\in\mathbb{R}^{n\times n\times n}, \textit{etc.}$$

As an example of tensor algebra multiplication, it holds that

$$g*g = c^2+2cv+cM+vv^\top+cT+v\otimes M+M\otimes v+\dots.$$

Note the difference between $v\otimes M$ and $M\otimes v$: it holds that $(v\otimes M)_{ijk} = v_iM_{jk}$, while $(M\otimes v)_{ijk} = v_kM_{ij}$.

[2]This is slightly non-standard notation: usually $\mathrm{T}(\mathcal{H})$ equals $\bigoplus_{M=0}^{\infty}\mathcal{H}^{\otimes M}$.

One checks that $\mathrm{T}(\mathcal{H})$ is indeed an algebra:

Proposition 3.9. $(\mathrm{T}(\mathcal{H}),+,*)$ *is a an associative* $\mathbb{R}$*-algebra with*

$$(1,0,\ldots,0,\ldots) \textit{ resp. } (0,\ldots,0,\ldots)$$

as multiplicative neutral element resp. additive neutral element. In general, the tensor algebra multiplication $*$ *is not commutative.*

Proof. Verifying the axioms of an associative $\mathbb{R}$-algebra is a series of non-trivial but elementary calculations. To see that $*$ is not commutative, consider the counter-example where $g,h \in \mathcal{H}$ are linearly independent. Then, $g \otimes h \neq h \otimes g$ (in the case of $\mathcal{H} = \mathbb{R}^n$, this is $gh^\top \neq hg^\top$). $\square$

Remark 3.10. *We further emphasize the following points, also to a reader who may already be familiar with the tensor product/outer product:*

(i) The Cartesian product $\mathcal{H}^M$ *is different from the tensor product* $\mathcal{H}^{\otimes M}$ *in the same way as* $(\mathbb{R}^n)^2$ *is different from* $\mathbb{R}^{n\times n} = (\mathbb{R}^n)^{\otimes 2}$ *and* $(\mathbb{R}^n)^3$ *from* $\mathbb{R}^{n\times n\times n} = (\mathbb{R}^n)^{\otimes 3}$.

(ii) In general, $*$ *is different from the formal tensor product/outer product of elements, since for* $g,h \in \mathrm{T}(\mathcal{H})$*, the formal tensor product* $g \otimes h$ *is an element of* $\mathrm{T}(\mathcal{H}) \otimes \mathrm{T}(\mathcal{H})$*, while the tensor power algebra product* $g * h$ *is an element of* $\mathrm{T}(\mathcal{H})$.

(iii) Under the identification introduced above, there is one case where $*$ *coincides with the tensor product - namely, when* g *and* h *are homogenous. Identifying* $g \in \mathcal{H}^{\otimes m}$ *and* $h \in \mathcal{H}^{\otimes n}$*, it holds that* $g \otimes h \in \mathcal{H}^{\otimes(m+n)}$ *may be identified with* $g * h$ *which is also homogenous. No equivalence of this kind holds when* g *and* h *are not homogenous.*

3.3. The signature as a canoncial feature set

We are now ready to define the signature features.

Definition 3.11. *We call the mapping*

$$\mathrm{S} : \mathrm{BV}([0,1],\mathcal{H}) \to \mathrm{T}(\mathcal{H}),\ x \mapsto \sum_{M \geq 0} \int_{\Delta^M([0,1])} (\mathrm{d}x)^{\otimes M}$$

the signature map of $\mathcal{H}$*-valued paths and we refer to* $\mathrm{S}(x)$*, as the signature features of* $x \in \mathrm{BV}$*. Similarly, we define (level-M-)truncated signature mapping as*

$$\mathrm{S}_{\leq M} : \mathrm{BV}([0,1],\mathcal{H}) \to \mathrm{T}(\mathcal{H}),\ x \mapsto \sum_{m=0}^{M} \int_{\Delta^m([0,1])} \mathrm{d}x^{\otimes M}.$$

Above is well-defined since the signature can be shown to have finite norm:

Lemma 3.12. *Let* $x \in \mathrm{BV}([0,1],\mathcal{H})$*. Then:*

(i) $\left\| \int_{\Delta^M[0,1]} \mathrm{d}x^{\otimes M} \right\|_{\mathcal{H}^{\otimes M}} \leq \frac{1}{M!} \mathrm{V}(x)^M < \infty,$

(ii) $\|\mathrm{S}(x)\|_{\mathrm{T}(\mathcal{H})} \leq \exp(\mathrm{V}(x)) < \infty.$

Proof. (i) is classical in the literature on bounded variation paths, it is also proven in Lemma B.4 of the appendix. (ii) follows from (i) by observing that $\|\mathrm{S}(x)\|_{\mathrm{T}(\mathcal{H})} \leq \sum_{M=0}^{\infty} \left\| \int_{\Delta^M[0,1]} \mathrm{d}x^{\otimes M} \right\|_{\mathcal{H}^{\otimes M}}$ due to the triangle equality, then substituting (i) and the Taylor expansion of exp. $\square$

There are several reasons why the signature features are (practically and theoretically) attractive, which we summarize before we state the results exactly:

1. the signature features are a **mathematically faithful representation of the underlying sequential data** x: the map $x \mapsto \mathrm{S}(x)$ is essentially one-on-one.
2. the signature features are **sequentially ordered analogues to polynomial features and moments**. The tensor algebra has the natural grading with M designating the "polynomial degree". It is further canonically compatible with natural operations on $\mathrm{BV}([0,1],\mathcal{H})$.
3. linear combinations of signature features **approximate continuous functions of sequential data arbitrarily well**. This is in analogy with classic polynomial features and implies that signature features are as rich a class for learning purposes as one can hope.

Theorem 1 (Signature uniqueness). *Let $x, y \in \mathrm{BV}([0,1],\mathcal{H})$. Then $\mathrm{S}(x) = \mathrm{S}(y)$ if and only x and y are equal up to tree-like equivalence*[3].

Proof. This is [20, Theorem 2.29 (ii)]. □

Remark 1. Being not tree-like equivalent is a very weak requirement, e.g. if x and y have a strictly increasing coordinate they are not tree-like equivalent. All the data we will encounter in the experiments is not tree-like. Even if presented with a tree-like path, simply adding time as extra coordinate (that is, working with $t \mapsto (t, x(t))$ instead of $t \mapsto x(t)$) guarantees the assumptions of above Theorem are met.

Remark 2. Above Theorem extends to unbounded variation paths, cf. [2]

Secondly, the signature features are analogous to polynomial features: the tensor algebra has a natural grading with M designating the "polynomial degree".

Theorem 2 (Chen's Theorem). *Let $x \in \mathrm{BV}([0,1],\mathcal{H})$, then*

$$\int_{\Delta^M} \mathrm{d}x^{\otimes M} \otimes \int_{\Delta^N} \mathrm{d}x^{\otimes N} = \sum_{\sigma} \sigma\left(\int_{\Delta^{M+N}} \mathrm{d}x^{\otimes(M+N)}\right).$$

Here the sum is taken over all ordered shuffles

$$\sigma \in \mathrm{OS}_{M,N} = \{\sigma : \sigma \text{ permutation of } \{1,\dots,M+N\}, \sigma(1) < \cdots < \sigma(M), \sigma(M+1) < \cdots < \sigma(M+N)\}.$$

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

Finally, a direct consequence of the above and again in analogy with classic polynomial features, linear combinations of signature features approximate continuous functions of sequential(!) data arbitrary well.

Theorem 3 (Linear approximations). *Let $\mathcal{P}$ be a compact subset of $\mathrm{BV}([0,1],\mathcal{H})$ of paths that are not tree-like equivalent. Let $f : \mathcal{P} \to \mathbb{R}$ be continuous in variation norm. Then for any $\epsilon > 0$, there exists a $w \in \mathrm{T}(\mathcal{H})$ such that*

$$\sup_{x \in \mathcal{P}} \left| f(x) - \langle w, \mathrm{S}(x) \rangle_{\mathrm{T}(\mathcal{H})} \right| < \epsilon.$$

Proof. The statement follows from the Stone–Weierstraß theorem if the set $\mathcal{F} \subset C(\mathcal{P},\mathbb{R})$

$$\mathcal{F} := \operatorname{span}\left\{\mathcal{P} \ni x \mapsto \left\langle e_{i_1} \otimes \cdots \otimes e_{i_M}, \mathrm{S}(x)\right\rangle_{\mathrm{T}(\mathcal{H})},\ M \in \mathbb{N}\right\} \tag{3.2}$$

forms a point-separating algebra. However, this is a direct consequence of the above: by Chen's theorem, Theorem 2, $\mathcal{F}$ is an algebra, and by the signature uniqueness, Theorem 1, $\mathcal{F}$ separates points. □

Above shows more than stated: for a fixed ONB (e_i) of $\mathrm{T}(\mathcal{H})$, there exists a finite subset of this ONB and w can be found in the linear span of this finite set.

[3]We call x, y tree-like equivalent if $x \sqcup y^{-1}$ is tree-like. A path $z \in \mathrm{BV}([a,b])$ is called tree-like if there exists a continuous map $h : [a,b] \to [0,\infty)$ with $h(0) = h(T) = 0$ and $|z(t) - (s)| \le h(s) + h(t) - 2\inf_{u \in [s,t]} h(u)$ for all $s \le t \in [a,b]$.

4. Kernelized signatures and sequentialized kernels

Our goal is to construct a kernel for sequential data of arbitrary type, to enable learning with such data. We proceed in two steps and first discuss the case when the sequential data are sequences in the Hilbert space $\mathcal{H}$ (for example $\mathcal{H}=\mathbb{R}^n$). In this scenario, the properties of the signature, presented in Section 3, suggest as kernel the scalar product of the signature features. This yields the following kernels,

Definition 4.1. *Fix $M \in \mathbb{N}$. We define the kernels*

$$\begin{aligned} \mathrm{K}^{\oplus} &: \mathrm{BV}(\mathcal{H}) \times \mathrm{BV}(\mathcal{H}) \to \mathbb{R}, \quad (x,y) \mapsto \langle \mathrm{S}(x), \mathrm{S}(y)\rangle_{\mathrm{T}(\mathcal{H})}, \\ \mathrm{K}^{\oplus}_{\leq M} &: \mathrm{BV}(\mathcal{H}) \times \mathrm{BV}(\mathcal{H}) \to \mathbb{R}, \quad (x,y) \mapsto \langle \mathrm{S}_{\leq M}(x), \mathrm{S}_{\leq M}(y)\rangle_{\mathrm{T}(\mathcal{H})}. \end{aligned}$$

We refer to $\mathrm{K}^{\oplus}$ *as the* signature kernel *and to* $\mathrm{K}^{\oplus}_{\leq M}$ *as the* signature kernel truncated at level M.

To make these kernels practically meaningful, we need to verify a number of points:

(a) That they are well-defined, positive (semi-)definite kernels. Note that checking finiteness of the scalar product is not immediate (but follows from well-known estimates about the signature features).

(b) That they are efficiently computable. A naive evaluation is infeasible, due to combinatorial explosion of the number of signature features. However, we show that $\mathrm{K}^{\oplus}$ and $\mathrm{K}^{\oplus}_{\leq M}$ can be expressed *entirely in integrals of inner products* $\langle \mathrm{d}x(s), \mathrm{d}y(t)\rangle_{\mathcal{H}}$. The formula can be written as an **efficient recursion**, similar to the Horner scheme for efficient evaluation of polynomials.

(c) That they are robust under discretization: the issue is that paths are never directly observed since all real observations are discrete sequences. The subsequent Section 5 introduces **discretizations for signatures**, and the two kernelization steps above, which are **canonical and consistent** to the above continuous steps, in a sampling sense of discrete sequences $\mathcal{H}^+$ converging to bounded variation path $\mathrm{BV}([0,1],\mathcal{H})$.

(d) That they are robust under noise: in most situations our measurements of the underlying path are perturbed by random perturbations and noise. We discuss the common situation of additive white noise/Brownian motion in Section 7.

We refer to the above procedure as "*kernel trick one*" and discuss it below in Section 4.1 and Section 4.2.

In a second step, to which we refer as "*kernel trick two*", we show that the above is also meaningful for sequential data in an arbitrary set $\mathcal{X}$. This second step yields, for any primary kernel k on (static objects) $\mathcal{X}$, a sequential kernel $\mathrm{k}^{\oplus}$ on (a sufficiently regular subset of) paths $\mathcal{P}(\mathcal{X}) \subset [[0,1] \to \mathcal{X}]$. We thus call this procedure that transforms a static kernel k on $\mathcal{X}$ into a kernel $\mathrm{k}^{\oplus}$ on sequences in $\mathcal{X}$, the so-called **sequentialization** (of the kernel k).

Definition 4.2. *Fix $M \in \mathbb{N}$ and* $\mathrm{k}: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\mathrm{k}(\cdot,\cdot) = \langle \phi, \phi\rangle_{\mathcal{H}}$. *We define*[4]

$$\begin{aligned} \mathrm{k}^{\oplus} &: \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}, \quad (\sigma,\tau) \mapsto \langle \mathrm{S}\left(\phi(\sigma)\right), \mathrm{S}\left(\phi(\tau)\right)\rangle_{\mathrm{T}(\mathcal{H})} \\ \mathrm{k}^{\oplus}_{\leq M} &: \mathcal{P}(\mathcal{X}) \times \mathcal{P}(\mathcal{X}) \to \mathbb{R}, \quad (\sigma,\tau) \mapsto \langle \mathrm{S}_{\leq M}\left(\phi(\sigma)\right), \mathrm{S}_{\leq M}\left(\phi(\tau)\right)\rangle_{\mathrm{T}(\mathcal{H})}. \end{aligned}$$

We refer to $\mathrm{k}^{\oplus}$ *as the* sequentialization *of the kernel* k *and to* $\mathrm{k}^{\oplus}_{\leq M}$ *as the* sequentialization *of the kernel* k *truncated at level M.*

As we have done for $\mathrm{K}^{\oplus}$ and $\mathrm{K}^{\oplus}_{\leq M}$, we again need to verify points (a), (b), (c), (d) for $\mathrm{k}^{\oplus}$ and $\mathrm{k}^{\oplus}_{\leq M}$. Point (a) follows under appropriate regularity assumptions on $\mathcal{P}(\mathcal{X})$ immediately from the corresponding statement for $\mathrm{K}^{\oplus}$ and $\mathrm{K}^{\oplus}_{\leq M}$. For point (b) note, that although the data enters $\mathrm{K}^{\oplus}$ and $\mathrm{K}^{\oplus}_{\leq M}$ in the recursion

[4]For $\phi: \mathcal{X} \to \mathcal{H}$ and $\sigma \in [[0,1] \to \mathcal{X}]$ we denote with $\phi(\sigma) \in [[0,1] \to \mathcal{H}]$ the path $t \mapsto \phi(\sigma(t))$.

formula only in the form of scalar products of differentials in $\mathcal{H}$, it is mathematically somewhat subtle to replace these by evaluations of k over an arbitrary set $\mathcal{X}$, due to the differential operators involved. However, we will do so by identifying the kernel k with a signed measure on $[0,1]^2$.

Points (a) and (b) will be discussed in this section, point (c) will be the main topic of Section 5, and point (d) is discussed in Section 7.

4.1. Well-definedness of the signature kernel

For (a) well-definedness, note: $\mathrm{K}^{\oplus}$ and $\mathrm{K}^{\oplus}_{\leq M}$ are positive definite kernels, since explicitly defined as a scalar product of features. Also, these scalar products are always finite (thus well-defined) for paths of bounded variation, as the following Lemma 4.3 shows.

Lemma 4.3. *Let $x, y \in \mathrm{BV}(\mathcal{H})$. Then it holds that*

(i) $\|\mathrm{K}^{\oplus}_{\leq M}(x,y)\| \leq \frac{1}{M!^2} V(x)^M V(y)^M < \infty$

(ii) $\|\mathrm{K}^{\oplus}(x,y)\| \leq \exp(V(x)+V(y)) < \infty$

Proof. This follows from Lemma 3.12 and the Cauchy–Schwarz-inequality. □

Hence, $\mathrm{K}^{\oplus}, \mathrm{K}^{\oplus}_{\leq M}$ are well-defined (positive definite) kernels.

4.2. Kernel trick number one: kernelizing the signature

The kernel trick consists of defining a kernel which is (b) efficiently computable. For this, we show that $\mathrm{K}^{\oplus}, \mathrm{K}^{\oplus}_{\leq M}$ can be entirely expressed in terms of $\mathcal{H}$-scalar products:

Proposition 4.4. *Let $x, y \in \mathrm{BV}(\mathcal{H})$. Then:*

(i) $\mathrm{K}^{\oplus}(x,y) = \sum_{m=0}^{\infty} \int_{(\boldsymbol{s},\boldsymbol{t})\in\Delta^m\times\Delta^m} \prod_{i=1}^{m} \langle \mathrm{d}x(\boldsymbol{s}[i]), \mathrm{d}y(\boldsymbol{t}[i])\rangle_{\mathcal{H}}$,

(ii) $\mathrm{K}^{\oplus}_{\leq M}(x,y) = \sum_{m=0}^{M} \int_{(\boldsymbol{s},\boldsymbol{t})\in\Delta^m\times\Delta^m} \prod_{i=1}^{m} \langle \mathrm{d}x(\boldsymbol{s}[i]), \mathrm{d}y(\boldsymbol{t}[i])\rangle_{\mathcal{H}}$.

Proof. The first formula follows from substituting definitions for $\mathrm{K}^{\oplus}_{\leq M}, \mathrm{K}^{\oplus}$ and using the linearity of the integrals (recall that we use the convention $\prod_{i=1}^{0} \cdots = 1$). □

Furthermore, there are the following Horner-scheme-type recursions:

Proposition 4.5. *Let $x, y \in \mathrm{BV}(\mathcal{H})$. Then*

(i) $\mathrm{K}^{\oplus}(x,y) = 1 + \int_{(s_1,t_1)\in(0,1)\times(0,1)} \left(1 + \int_{(s_2,t_2)\in(0,s_1)\times(0,t_1)} (1+\cdots)\cdots \langle \mathrm{d}x(s_2), \mathrm{d}y(t_2)\rangle_{\mathcal{H}}\right) \langle \mathrm{d}x(s_1), \mathrm{d}y(t_1)\rangle_{\mathcal{H}}$,

(ii) $\mathrm{K}^{\oplus}_{\leq M}(x,y) = 1 + \int_{(s_1,t_1)\in(0,1)\times(0,1)} \left(1 + \ldots \int_{(s_M,t_M)\in(0,s_{M-1})\times(0,t_M-1)} \langle \mathrm{d}x(s_M), \mathrm{d}y(t_M)\rangle_{\mathcal{H}} \ldots\right) \langle \mathrm{d}x(s_1), \mathrm{d}y(t_1)\rangle_{\mathcal{H}}$.

Proof. This follows from Proposition 4.4 and an elementary computation. □

The recursion is the mathematically precise form of the iterated expectation from Section 1.3. In Section 5 we show that this recursion is preserved under discretization.

4.3. Kernel trick number two: sequentialization of kernels

As shown in the previous Section 4.2, the signature kernel on bounded variation paths over $\mathcal{H}$ can be entirely expressed in scalar products over $\mathcal{H}$. However, in general, the data will not be directly observed as sequences in a (potentially infinite dimensional) Hilbert space $\mathcal{H}$, but in some arbitrary $\mathcal{X}$. Sequences in $\mathcal{X}$ can be treated with a *second* kernelization step: We choose a primary feature map $\phi : \mathcal{X} \to \mathcal{H}$ for (non-sequential) data in $\mathcal{X}$. By Proposition 4.4, the sequential kernel can be expressed in scalar products in $\mathcal{H}$, therefore, after the second kernelization step, in the primary kernel.

We make this point precise.

Assumption 4.6. *For the general situation of sequences in an arbitrary set $\mathcal{X}$, we may assume:*

(i) *The potential observations, denoted $\mathcal{P}(\mathcal{X})$, are sequential data of type $[[0,1] \to \mathcal{X}]$.*

(ii) *The potential observations $\mathcal{P}(\mathcal{X})$ are absolutely continuous paths when corresponding sequences of primary features are considered. That is, for every potential observation $\sigma \in \mathcal{P}(\mathcal{X})$, the concatenation $\phi \circ \sigma$ is an element of* BV($\mathcal{H}$) *that is absolutely continuous*[5].

(iii) *Scalar products of primary features can be efficiently computed via a kernel function*

$$\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad \mathrm{k}(x,y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}.$$

More precisely, $\mathcal{H}$ is the RKHS associated to the kernel k.

Under Assumption 4.6, the formulas given in Proposition 4.4 and Proposition 4.5 suggest to substitute $x = \phi(\sigma), y = \phi(\tau)$ and to replace $\langle \mathrm{d}x(\boldsymbol{s}[i]), \mathrm{d}y(\boldsymbol{t}[i]) \rangle_{\mathcal{H}}$ with an evaluation of k. However, the differential is between the scalar product and the ϕ and hence a naive replacement of the scalar product by the primary kernel k is not possible. Below we show that a slightly more technical form of the iterated expectation formula in the introductory Section 1.2 holds and agrees with this formula for differentiable paths:

Proposition 4.7. *Let Assumptions 4.6 be satisfied. Then $\mathrm{k}^{\oplus}$ and $\mathrm{k}^{\oplus}_{\leq M}$ are positive definite kernels on $\mathcal{P}(\mathcal{X})$ and for $\sigma, \tau \in \mathcal{P}(\mathcal{X})$:*

(i) $\mathrm{k}^{\oplus}(\sigma,\tau) = \sum_{m=0}^{\infty} \int_{(\boldsymbol{s},\boldsymbol{t}) \in \Delta_m \times \Delta_m} \mathrm{d}\kappa_{\sigma,\tau}(\boldsymbol{s}[1], \boldsymbol{t}[1]) \cdot \ldots \cdot \mathrm{d}\kappa_{\sigma,\tau}(\boldsymbol{s}[m], \boldsymbol{t}[m]),$

(ii) $\mathrm{k}^{\oplus}_{\leq M}(\sigma,\tau) = \sum_{m=0}^{M} \int_{(\boldsymbol{s},\boldsymbol{t}) \in \Delta_m \times \Delta_m} \mathrm{d}\kappa_{\sigma,\tau}(\boldsymbol{s}[1], \boldsymbol{t}[1]) \cdot \ldots \cdot \mathrm{d}\kappa_{\sigma,\tau}(\boldsymbol{s}[m], \boldsymbol{t}[m]).$

for a suitably chosen signed measure $\kappa_{\sigma,\tau}$ on $[0,1]^2$. If $\mathcal{X}$ is an $\mathbb{R}$-vector space and σ, τ are differentiable, then

$$\mathrm{d}\kappa_{\sigma,\tau}(s,t) = \mathrm{k}(\dot{\sigma}(s), \dot{\tau}(t)) \mathrm{d}s \, \mathrm{d}t.$$

Proof. The proof is carried out in Appendix A. □

As before, above can be written as Horner-type recursions

Proposition 4.8. *Let Assumptions 4.6 be satisfied and $\sigma, \tau \in \mathcal{P}(\mathcal{X})$. Then:*

(i) $\mathrm{k}^{\oplus}(\sigma,\tau) = 1 + \int_{(s_1,t_1) \in (0,1) \times (0,1)} \left(1 + \int_{(s_2,t_2) \in (0,s_1) \times (0,t_1)} (1 + \cdots) \cdots \mathrm{d}\kappa_{\sigma,\tau}(s_2,t_2)\right) \mathrm{d}\kappa_{\sigma,\tau}(s_1,t_1)$

(ii) $\mathrm{k}^{\oplus}_{\leq M}(\sigma,\tau) = 1 + \int_{(s_1,t_1) \in (0,1) \times (0,1)} \left(1 + \ldots \int_{(s_M,t_M) \in (0,s_{M-1}) \times (0,t_M - 1)} \mathrm{d}\kappa_{\sigma,\tau}(s_1,t_2) \ldots \right) \mathrm{d}\kappa_{\sigma,\tau}(s_1,t_2)$

Proof. This follows from Proposition 4.7 and an elementary calculation. □

[5] Absolutely continuous paths are dense in variation norm in BV. We exclude non-absolutely continuous paths (like Cantor functions) to avoid technicalities. In Section 7 we show that above can generalized to much "rougher paths" than bounded variation.

5. Discrete signatures and kernels for sequences

As it was mentioned under point (c) in Section 4, there is one major practical issue with the kernels introduced in Section 4: continuous sequences are in reality never fully observed, since observations in practice are always finite, for example given as time points at which a time series is sampled. To be more precise, instead of having full knowledge of $\sigma \in [[0,1] \to \mathcal{X}]$ we can use at most a finite number of samples, $\sigma(\boldsymbol{s}) = \big(\sigma(s_1), \ldots, \sigma(s_n)\big) \in \mathcal{H}^+$ for $\boldsymbol{s} \in \Delta$. We address this by providing a discretization of all prior concepts: a discrete version $\mathfrak{S} : \mathcal{H}^+ \to \mathrm{T}(\mathcal{H})$ of the signature S, a discrete version $\mathrm{K}^+ : \mathcal{H}^+ \times \mathcal{H}^+ \to \mathbb{R}$ of the signature kernel $\mathrm{K}^{\oplus}$, and a discrete version $\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+$ of the sequnentialization $\mathrm{k}^{\oplus}$ of k.

The discretization is based on an elementary integral-vs-sum approximation argument, in which

$$\text{the iterated integral } \int_{\Delta^m([0,1])} \mathrm{d}x^{\otimes m} \text{ is approximated by a sum } \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\nabla \boldsymbol{x})] \\ \ell(\boldsymbol{i}) = m}} \nabla \boldsymbol{x}[i_1] \otimes \cdots \otimes \nabla \boldsymbol{x}[i_m],$$

where $\boldsymbol{x} = x(\boldsymbol{t})$ is a suitable discretization of the bounded variation path $x \in \mathrm{BV}([0,1], \mathcal{H})$ with $\boldsymbol{t} \in \Delta([0,1])$. Both the integral and the sum live in the tensor power algebra, an we will show that the sum approximates the integral in a sampling sense. Similarly, one can define a signature for discrete sequences that approximates the continuous signature. The beginning of this section showcases qualitative statements of this approximative kind.

In analogy to Section 4 we proceed in two steps: in the first step, "*kernel trick one*", we use the discretizted signature $\mathfrak{S}$ to define a kernel K^+ for sequences in $\mathcal{H}^+$ (of possibly different length). It can be expressed as a polynomial in scalar products in $\mathcal{H}$, as follows[6]:

$$\mathrm{K}^+(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\nabla \boldsymbol{x})] \\ \boldsymbol{j} \sqsubset [\ell(\nabla \boldsymbol{y})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{i})}} \prod_{r=1}^{\ell(\boldsymbol{i})} \langle \nabla \boldsymbol{x}[i_r], \nabla \boldsymbol{y}[j_r] \rangle_{\mathcal{H}} \tag{5.1}$$

and also a Horner-type formula is derived (see Proposition 5.5 below). In a second step, "*kernel trick two*", we replace the inner product of finite differences in (5.1) by a linear combination of evaluations of the kernel k. This gives a kernel k^+ defined on sequence in $\mathcal{X}$. Primary kernels k on arbitrary sets $\mathcal{X}$ can thus be "sequentialized" to kernels on sequences in such sets.

Besides the Horner type formula, the main theoretical result of this section is that both kernels are robust in the sense that $\mathrm{K}^+(x(\boldsymbol{s}), y(\boldsymbol{t}))$ and $\mathrm{k}^+(\sigma(\boldsymbol{s}), \tau(\boldsymbol{t}))$ converge to $\mathrm{K}^{\oplus}(x, y)$ and $\mathrm{k}^{\oplus}(\sigma, \tau)$ as the mesh of $\boldsymbol{s}$ and $\boldsymbol{t}$ vanishes.

In Section 6 we will also see that the well-known string kernels can be derived as such a special case of sequentialization, for sequences of symbols; in section 7 we show that above kernels are also robust under noise, that is when the bounded variation assumption does not hold; in Section 8, we will further show that despite an exponential number of terms in the sum-product expansion above, the sequential kernels can be computed efficiently.

5.1. Discretizing the signature

To obtain a kernel K^+ for discrete sequences in $\mathcal{H}^+$ that mimicks the signature kernel $\mathrm{K}^{\oplus}$, we pass from the Riemann–Stieltjes integral to a discretized, finite-sum approximation. The central mathematical observation is that

$$\bigotimes_{j=0}^{\ell(\nabla \boldsymbol{x})} (1 + \nabla \boldsymbol{x}[j]) = \sum_{\boldsymbol{i} \sqsubset [\ell(\nabla \boldsymbol{x})]} \nabla \boldsymbol{x}[i_1] \otimes \cdots \otimes \nabla \boldsymbol{x}[i_{\ell(\boldsymbol{i})}] \approx \sum_{m=0}^{\infty} \int_{\Delta^m([0,1])} \mathrm{d}x^{\otimes m},$$

[6]Recall our convention $\prod_{r=1}^{0} f(r) = 1$ and that $\boldsymbol{i} \sqsubset [L]$ includes the empty tuple.

that is, the sum approximating the signature can be written as an outer product reminiscent of an exponential approximation.

The mathematical statement underlying our approximation generalizes Euler's famous theorem on the exponential function. Euler's original theorem and its proof are below, in a form that is slightly more quantitative than how it is usually presented, while being closer to our later approximation result:

Theorem 4. *Let $x \in \mathbb{R}, n \in \mathbb{N}$. Then,*

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

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. $\square$

Approximation bounds for a discretized version of the signature approximating the continuous one will be given by a generalization of Euler's Theorem 4. We introduce a discretized variant of the signature for sequences, as follows:

Definition 5.1. *We call the map $\mathfrak{S} : \mathcal{H}^+ \to \mathrm{T}(\mathcal{H})$ defined as $\mathfrak{S}(\boldsymbol{x}) = \prod_{i=1}^{\ell(\nabla \boldsymbol{x})}(1+\nabla\boldsymbol{x}[i])$ the (approximate) signature map, for discrete sequences.*

We are ready to prove the main theorem for discrete approximation of the signature. The gist of Theorem 5 is that the signature of sub-sequences of x approximates the actual signature of x, in a limit similar to Euler's Theorem 4.

Theorem 5. *Let $x \in \mathrm{BV}(\mathcal{H})$ and $\boldsymbol{t} \in \Delta^{M+1}([0,1])$. Then,*

$$\|\mathfrak{S}(x(\boldsymbol{t})) - \mathrm{S}(x)\|_{\mathrm{T}(\mathcal{H})} \leq \exp(\mathrm{V}(x)) - \exp_1(\mathrm{V}_{\boldsymbol{t}}(x)),$$

where $\exp(\boldsymbol{r}) := \prod_{i=1}^{\ell(\boldsymbol{r})}(1+\boldsymbol{r}[j])$ for $\boldsymbol{r} \in \mathbb{R}^+$, and $\mathrm{V}_{\boldsymbol{t}}(x) := \big(\mathrm{V}(x[t_1,t_2]),\ldots,\mathrm{V}(x[t_M,t_{M+1}])\big) \in \mathbb{R}^M$ (as usual, $x[a,b]$ denotes the restriction of x to $\mathrm{BV}([a,b],\mathcal{H})$.). Further, it holds that:

(i) The right hand sides are always positive and can be bounded as follows:

$$0 \leq \exp(\mathrm{V}(x)) - \exp_1(\mathrm{V}_{\boldsymbol{t}}(x)) \leq \mathrm{V}(x)\exp(\mathrm{V}(x))\cdot\|\mathrm{V}_{\boldsymbol{t}}(x)\|_0,$$

where as usual we denote $\|\mathrm{V}_{\boldsymbol{t}}(x)\|_0 = \max_i \mathrm{V}\big(x[t_i,t_{i+1}]\big)$.

(ii) One has convergence on the right side

$$\lim_{\mathrm{V}(x(\boldsymbol{t}))\to\mathrm{V}(x)} \exp(\mathrm{V}_{\boldsymbol{t}}(x)) = \exp(\mathrm{V}(x))$$

(and similar for $\exp_M$*), uniform of order* $O\left(\|\mathrm{V}_{\boldsymbol{t}}(x)\|_0\right)$ *on any compact subset of* $\mathrm{BV}(\mathcal{H})$*.*

(iii) In particular, one has convergence of the discretized signature to the continuous signature

$$\lim_{\mathrm{V}(x(\boldsymbol{t}))\to\mathrm{V}(x)} \mathfrak{S}(x(\boldsymbol{t})) = \mathrm{S}(x),$$

(hence also for $\mathrm{S}_{\leq M}$*), uniform of order* $O\left(\|\mathrm{V}_{\boldsymbol{t}}(x)\|_0\right)$ *on any compact subset of* $\mathrm{BV}(U)$*.*

(iv) If $\boldsymbol{t}$ *is chosen such that* $\mathrm{V}(x[t_i, t_{i+1}]) = \frac{\mathrm{V}(x)}{M}$ *for all i, then one has the asymptotically tight bound*

$$\|\mathfrak{S}(x(\boldsymbol{t})) - \mathrm{S}(x)\|_{\mathrm{T}(\mathcal{H})} \leq \frac{\exp(\mathrm{V}(x))}{M}\left(1 + \frac{(\mathrm{V}x)^M}{(M-2)!}\right).$$

Proof. The proof is carried out in Appendix B: The main statement follows from Theorem 6 (ii); points (i) and (ii) follow from applying Proposition B.6 to $\mathrm{V}(x) = \sum_{i=1}^M \mathrm{V}(x[t_i, t_{i+1}])$; point (iii) follows from (ii); (iv) follows from Theorem 6 (iii). ☐

Euler's original Theorem 4 is recovered from for substituting $x : I \to \mathbb{R}, t \mapsto x' \cdot t$ in Theorem 5 (iii), where x' is the x of Theorem 4. Similar statements for the truncated signature may be derived, where the bounds are truncated versions of the ones given; for an exact mathematical formulation, see Theorem 6 (i).

Remark 5.2 (About geometric approximations)**.** *In general, there is no bounded variation path* $x' \in \mathrm{BV}(\mathcal{H})$ *such that* $\mathrm{S}(x') = \mathfrak{S}(x(\boldsymbol{t}))$*, even though both* $\mathfrak{S}(x(\boldsymbol{t}))$ *and* $\mathrm{S}(x)$ *are elements of the tensor algebra* $\mathrm{T}(\mathcal{H})$*. Thus the approximation characterized in Theorem 5 is an algebraic, non-geometric approximation on the level of signatures (= algebraic objects); since, in general, it does not arise from an approximation/discretization on the level of paths (= geometric objects).*

The latter is the common type of approximation for (rough) paths, see [8]. The reader familiar with such approximations on the level of paths is invited to follow the exposition in parallel with any path-level approximation in mind. The theoretical considerations can be carried out in analogy for a while, but to our knowledge do not lead to a kernel which is as efficiently computable (as opposed to the efficient Algorithm 3 discussed in Section 8); see also Remark 8.1 about approximations of inner products of signature features of piecewise linear paths.

5.2. Kernel trick number one discretized

In the discrete setting, we have access only to the discrete signature $\mathfrak{S}(x(\boldsymbol{t}))$ of a path x, at a finite sequence of points $\boldsymbol{t} \in \Delta$. Theorem 5 guarantees that $\mathfrak{S}(x(\boldsymbol{t}))$ is a good approximation of the continuous signature $\mathrm{S}(x)$ when $\boldsymbol{t}$ is densely sampled (as $\mathrm{V}(x(\boldsymbol{t}))$ approaches $\mathrm{V}(x)$).

Both signatures live in the tensor algebra, it is therefore natural to obtain discretized variant of signature kernels as the scalar product of discretized signatures; again, we also define a truncated version.

Definition 5.3. *The discretized signature kernel* K^+*, for sequences in* $\mathcal{H}$*, and its truncation* K^+_M*, are defined as*

$$\begin{aligned} \mathrm{K}^+ &: \mathcal{H}^+ \times \mathcal{H}^+ \to \mathbb{R}, \quad (\boldsymbol{x}, \boldsymbol{y}) \mapsto \langle \mathfrak{S}(\boldsymbol{x}), \mathfrak{S}(\boldsymbol{y})\rangle_{\mathrm{T}(\mathcal{H})} \\ \mathrm{K}^+_M &: \mathcal{H}^+ \times \mathcal{H}^+ \to \mathbb{R}, \quad (\boldsymbol{x}, \boldsymbol{y}) \mapsto \langle \mathfrak{S}_M(\boldsymbol{x}), \mathfrak{S}_M(\boldsymbol{y})\rangle_{\mathrm{T}(\mathcal{H})}. \end{aligned}$$

Both K^+ and K^+_M are a positive (semi-)definite kernels, since explicitly defined as a scalar product of features.

Through the approximation Theorem 5, the sequential kernel naturally applies to a ground truth of bounded variation paths $x, y \in \mathrm{BV}([0,1], \mathcal{H})$ which is sampled in a finite number of consecutive points $\boldsymbol{s}, \boldsymbol{t} \in \Delta([0,1])$, by considering $\mathrm{K}^+(x(\boldsymbol{s}), y(\boldsymbol{t}))$ (note that $\boldsymbol{s}$ and $\boldsymbol{t}$ being of different lengths will create no issue in-principle). We obtain a corollary of Theorem 5 for approximating the kernel function in this way:

Corollary 5.4. *Let $x, y \in \mathrm{BV}([0,1], \mathcal{H})$, let $\boldsymbol{s}, \boldsymbol{t} \in \Delta([0,1])$. Then,*

$$\left\|\mathrm{K}^+(x(\boldsymbol{s}), y(\boldsymbol{t})) - \mathrm{K}^\oplus(x,y)\right\| \le 4\exp(\mathrm{V}(x)+\mathrm{V}(y)) - 2\exp(\mathrm{V}(y))\exp(\mathrm{V}_{\boldsymbol{s}}(x)) - 2\exp(\mathrm{V}(x))\exp(\mathrm{V}_{\boldsymbol{t}}(y)).$$

In particular, it holds that

$$\lim_{\substack{\mathrm{V}(x(\boldsymbol{s}))\to\mathrm{V}(x) \\ \mathrm{V}(y(\boldsymbol{t}))\to\mathrm{V}(y)}} \mathrm{K}^+(x(\boldsymbol{s}), y(\boldsymbol{t})) = \mathrm{K}^\oplus(x,y),$$

where convergence is uniform of order $O(\|\mathrm{V}_{\boldsymbol{s}}(x)\|_0 + \|\mathrm{V}_{\boldsymbol{t}}(y)\|_0)$ on any compact subset of $\mathrm{BV}([0,1],\mathcal{H}) \times \mathrm{BV}([0,1],\mathcal{H})$.

Proof. It holds that

$$\begin{aligned} &\mathrm{K}^+(x(\boldsymbol{s}), y(\boldsymbol{t})) - \mathrm{K}^\oplus(x,y) \\ =&\langle \mathfrak{S}(x(\boldsymbol{s})), \mathfrak{S}(y(\boldsymbol{t}))\rangle_{\mathrm{T}(\mathcal{H})} - \langle \mathrm{S}(x), \mathrm{S}(y)\rangle_{\mathrm{T}(\mathcal{H})} \\ =&\langle \mathfrak{S}(x(\boldsymbol{s})), \mathfrak{S}(y(\boldsymbol{t})) - \mathrm{S}(y)\rangle_{\mathrm{T}(\mathcal{H})} + \langle \mathfrak{S}(x(\boldsymbol{s})) - \mathrm{S}(x), \mathrm{S}(y)\rangle_{\mathrm{T}(\mathcal{H})}. \end{aligned}$$

The Cauchy-Schwarz-inequality implies that

$$\|\langle \mathfrak{S}(x(\boldsymbol{s})), \mathfrak{S}(y(\boldsymbol{t})) - \mathrm{S}(y)\rangle_{\mathrm{T}(\mathcal{H})}\|_{\mathrm{T}(\mathcal{H})} \le \|\mathfrak{S}(x(\boldsymbol{s}))\| \cdot \|\mathfrak{S}(y(\boldsymbol{t})) - \mathrm{S}(y)\|$$

Theorem 5, together with Lemma 3.12 (i), implies that

$$\|\mathfrak{S}(x(\boldsymbol{s}))\| \cdot \|\mathfrak{S}(y(\boldsymbol{t})) - \mathrm{S}(y)\| \le 2\exp(\mathrm{V}(x)) \cdot \left(\exp(\mathrm{V}(y)) - \exp(\mathrm{V}_{\boldsymbol{t}}(y))\right)$$

Similarly, one obtains

$$\|\langle \mathfrak{S}(y(\boldsymbol{t})), \mathfrak{S}(x(\boldsymbol{s})) - \mathrm{S}(x)\rangle_{\mathrm{T}(\mathcal{H})}\|_{\mathrm{T}(\mathcal{H})} \le 2\exp(\mathrm{V}(y)) \cdot \left(\exp(\mathrm{V}(x)) - \exp(\mathrm{V}_{\boldsymbol{s}}(x))\right).$$

Putting all (in-)equalities together yields the main claim, the convergence statement follows from Theorem 5 (ii). ☐

Note that the sampling points $\boldsymbol{s}, \boldsymbol{t}$ are crucial for the convergence statement, even though the definition of $\mathrm{K}^\oplus$ itself depends only on x, y. A similar statement may be obtained for K^+_M in analogy.

We proceed by giving explicit formulae for K^+ that will become crucial for its efficient computation.

Proposition 5.5. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}^+$. The following identities hold for the discretized signature kernels:*

(i) $\mathrm{K}^+(\boldsymbol{x},\boldsymbol{y}) = \sum_{\substack{\boldsymbol{x}'\sqsubset\nabla\boldsymbol{x}, \boldsymbol{y}'\sqsubset\nabla\boldsymbol{y} \\ \ell(\boldsymbol{x}')=\ell(\boldsymbol{y}')}} \prod_{i=1}^{\ell(\boldsymbol{x}')} \langle \boldsymbol{x}'[i], \boldsymbol{y}'[i]\rangle_{\mathcal{H}}$,

(ii) $\mathrm{K}^+_M(\boldsymbol{x},\boldsymbol{y}) = \sum_{\substack{\boldsymbol{x}'\sqsubset\nabla\boldsymbol{x}, \boldsymbol{y}'\sqsubset\nabla\boldsymbol{y} \\ \ell(\boldsymbol{x}')=\ell(\boldsymbol{y}')\le M}} \prod_{i=1}^{\ell(\boldsymbol{x}')} \langle \boldsymbol{x}'[i], \boldsymbol{y}'[i]\rangle_{\mathcal{H}}$

(iii) $\mathrm{K}^+(\boldsymbol{x},\boldsymbol{y}) = 1 + \sum_{\substack{i_1\ge 1 \\ j_1\ge 1}} \langle \nabla\boldsymbol{x}[i_1], \nabla\boldsymbol{y}[j_1]\rangle \left(1 + \sum_{\substack{i_2\gtrsim i_1 \\ j_2\gtrsim j_1}} \langle \nabla\boldsymbol{x}[i_2], \nabla\boldsymbol{y}[j_2]\rangle \left(1 + \sum_{\substack{i_3\gtrsim i_2 \\ j_3\gtrsim j_2}} \langle \nabla\boldsymbol{x}[i_3], \nabla\boldsymbol{y}[j_3]\rangle \left(1 + \sum_{\dots} \dots\right)\right)\right)$

(iv) $\mathrm{K}^+_M(\boldsymbol{x},\boldsymbol{y}) = 1 + \sum_{\substack{i_1\ge 1 \\ j_1\ge 1}} \langle \nabla\boldsymbol{x}[i_1], \nabla\boldsymbol{y}[i_1]\rangle \left(1 + \sum_{\substack{i_2\gtrsim i_1 \\ j_2\gtrsim j_1}} \langle \nabla\boldsymbol{x}[i_2], \nabla\boldsymbol{y}[i_2]\rangle \left(1 + \cdots \sum_{\substack{i_M\gtrsim i_{M-1} \\ j_M\gtrsim j_{M-1}}} \langle \nabla\boldsymbol{x}[i_M], \nabla\boldsymbol{y}[i_M]\rangle\right)\right)$

where the usual convention that a sum running over an empty index set evaluates to zero applies.

Proof. This follows directly from an explicit computation where both sides are expanded and compared. □

Remark 5.6. *All summation in Proposition 5.5 is finite, even for* K^+*: there are only a finite number of sub-sequences* $\boldsymbol{x}', \boldsymbol{y}'$ *in (i); and in (ii), summation ends at* $M = \min(\ell(\boldsymbol{x}), \ell(\boldsymbol{y})) - 1$. *An important feature of the sum-formula given by Proposition 5.5 (iv) is that it is more efficient to evaluate than the more naive presentation in Proposition 5.5 (ii), due to a much smaller amount of summation.*

5.3. Kernel trick number two discretized

Both equalities in Proposition 5.5 show K^+ to be expressible entirely in terms of scalar products in $\mathcal{H}$. Thus, if we are again in the situation of Assumptions 4.6, that is there is a primary kernel function $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that $\mathrm{k}(\sigma, \tau) = \langle \phi(\sigma), \phi(\tau) \rangle_{\mathcal{H}}$ for $\sigma, \tau \in \mathcal{X}$, the sequential kernel can be again be second-kernelized

Definition 5.7. *Let* $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. *The (discrete) sequentialization* k^+ *of* k *for sequences in* $\mathcal{X}$, *and its truncation* k^+_M *are defined as*

$$\begin{aligned} \mathrm{k}^+ &: \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad (\boldsymbol{\sigma}, \boldsymbol{\tau}) \mapsto \langle \mathfrak{S}(\phi(\boldsymbol{\sigma})), \mathfrak{S}(\phi(\boldsymbol{\tau})) \rangle_{\mathrm{T}(\mathcal{H})} \\ \mathrm{k}^+_M &: \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad (\boldsymbol{\sigma}, \boldsymbol{\tau}) \mapsto \langle \mathfrak{S}_M(\phi(\boldsymbol{\sigma})), \mathfrak{S}_M(\phi(\boldsymbol{\tau})) \rangle_{\mathrm{T}(\mathcal{H})}. \end{aligned}$$

Following the presentation in the previous section 5.1 and making the substitution $x = \phi(\sigma)$, $y = \phi(\tau)$ yields again an explict sum formula and a recursive formula for k^+ and k^+_M, in direct analogy to Proposition 5.5. This discrete recursion is at the foundation of efficient computation in a practical setting, as demonstrated in Section 6 and Section 8.

Proposition 5.8. *Let* $\boldsymbol{\sigma}, \boldsymbol{\tau} \in \mathcal{X}^+$ *and* $M \in \mathbb{N}$. *The following identities hold for the sequentialization* k^+ *of* k*:*

(i) $\mathrm{k}^+(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\nabla \boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\nabla \ell(\boldsymbol{\tau})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j})}} \prod_{r=1}^{\ell(\boldsymbol{i})} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[\boldsymbol{i}[r], \boldsymbol{j}[r]]$

(ii) $\mathrm{k}^+(\boldsymbol{\sigma}, \boldsymbol{\tau}) = 1 + \sum_{\substack{i_1 \geq 1 \\ j_1 \geq 1}} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i_1, j_1] \cdot \left(1 + \sum_{\substack{i_2 \gtrsim i_1 \\ j_2 \gtrsim j_1}} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i_2, j_2] \cdot \left(1 + \sum_{\dots} \cdots \right) \right),$

(iii) $\mathrm{k}^+_M(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\nabla \boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\ell(\nabla \boldsymbol{\tau})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j}) \leq M}} \prod_{r=1}^{\ell(\boldsymbol{i})} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[\boldsymbol{i}[r], \boldsymbol{j}[r]],$

(iv) $\mathrm{k}^+_M(\boldsymbol{\sigma}, \boldsymbol{\tau}) = 1 + \sum_{\substack{i_1 \geq 1 \\ j_1 \geq 1}} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i_1, j_1] \cdot \left(1 + \sum_{\substack{i_2 \gtrsim i_1 \\ j_2 \gtrsim j_1}} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i_2, j_2] \cdot \left(1 + \cdots \sum_{\substack{i_M \gtrsim i_{M-1} \\ j_M \gtrsim j_{M-1}}} \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i_M, j_M] \right) \right).$

where we use the notation

$$\nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i, j] := \mathrm{k}(\boldsymbol{\sigma}[i+1], \boldsymbol{\tau}[j+1]) + \mathrm{k}(\boldsymbol{\sigma}[i], \boldsymbol{\tau}[j]) - \mathrm{k}(\boldsymbol{\sigma}[i], \boldsymbol{\tau}[j+1]) - \mathrm{k}(\boldsymbol{\sigma}[i+1], \boldsymbol{\tau}[j]).$$

Proof. Substituting $\boldsymbol{x} = \phi(\boldsymbol{\sigma}), \boldsymbol{y} = \phi(\boldsymbol{\tau})$ we have for $\boldsymbol{i} = (i_1, \dots, i_r), \boldsymbol{j} = (j_1, \dots, j_r)$

$$\langle (\nabla \boldsymbol{x})[i_r], (\nabla \boldsymbol{y})[j_r] \rangle_{\mathcal{H}} = \langle (\nabla \phi(\boldsymbol{\sigma}))[i_r], (\nabla \phi(b\tau))[j_r] \rangle_{\mathcal{H}}$$

for the scalar products. One obtains, via an elementary computation, that

$$\langle (\nabla \boldsymbol{x})[i_r], (\nabla \boldsymbol{y})[j_r] \rangle_{\mathcal{H}} = \nabla_{i_r, j_r} \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau}).$$

The statement then follows by Proposition 5.5 □

Remark 5.9. *Again a direct consequence of the approximation Theorem 5 is the convergence of* k^+ *to* $\mathrm{k}^{\oplus}$,

$$\mathrm{k}^+(\sigma(\boldsymbol{s}),\tau(\boldsymbol{t})) \to \mathrm{k}^{\oplus}(\sigma,\tau) \text{ as } \operatorname{mesh}(\boldsymbol{s}),\operatorname{mesh}(\boldsymbol{t}) \text{ vanishes}$$

(under Assumptions 4.6). However, although the rate of convergence is explicitly given by Theorem 5, it depends on $\mathrm{V}(\phi(\sigma))$ *and* $\mathrm{V}(\phi(\tau))$.

Remark 5.10 (Variations on a theme). *We further point out several, more complex variants of the second kernelization:*

(i) *In the formula for the sequential kernel, one could also simply omit the first differences. This corresponds to cumulative summation of the sequences in feature space, or replacing signatures by exponentials:*

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (\boldsymbol{\sigma},\boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{\sigma}' \sqsubset \boldsymbol{\sigma},\, \boldsymbol{\tau}' \sqsubset \boldsymbol{\tau} \\ \ell(\boldsymbol{\sigma}')=\ell(\boldsymbol{\tau}')}} \prod_{r=1}^{\ell(\boldsymbol{\sigma}')} \mathrm{k}(\boldsymbol{\sigma}'[r],\boldsymbol{\tau}'[r]).$$

In practice, we have anecdotally found that this to lead to large sums and numerical instabilities — though one could argue that the variant without finite differences is the simpler one.

(ii) *More generally, one may consider a family of primary kernels, of form* $\mathrm{k} : \mathcal{X}^m \times \mathcal{X}^m \to \mathbb{R}$, *not necessarily induced as the product of kernels of type* $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, *sequentialization is possible via Proposition 5.5 (i), namely as*

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (\boldsymbol{\sigma},\boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})],\, \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})] \\ \ell(\boldsymbol{i})=\ell(\boldsymbol{j})=m}} \mathrm{k}^m(\boldsymbol{\sigma}[\boldsymbol{i}],\boldsymbol{\tau}[\boldsymbol{j}]).$$

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

(iii) *One may additionally opt to have the primary kernel remember the position of elements in the sequence. This may be done by mapping sequences in* $\mathcal{X}^+$ *to a position-remembering sequence in* $(\mathcal{X} \times \mathbb{N})^+$ *first, and then sequentializing a primary kernel which is a product kernel of type* $\mathrm{k}^m((\boldsymbol{\sigma},\boldsymbol{i}),(\boldsymbol{\tau},\boldsymbol{j})) = \kappa(\boldsymbol{i},\boldsymbol{j}) \cdot \mathrm{k}^m(\boldsymbol{\sigma},\boldsymbol{\tau})$, *where* $\kappa : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{R}$ *is positive definite. This yields a sequential kernel of form*

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (\boldsymbol{\sigma},\boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})],\, \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})] \\ \ell(\boldsymbol{i})=\ell(\boldsymbol{j})}} \kappa(\boldsymbol{i},\boldsymbol{j}) \cdot \mathrm{k}^m(\boldsymbol{\sigma}[\boldsymbol{i}],\boldsymbol{\tau}[\boldsymbol{j}]).$$

From the viewpoint of the sequential kernel, the above choices, while minor, seem somewhat arbitrary. We will see in the coming Section 6 that they have their justification in explaining prior art. Nevertheless we would with Occam's razor that they should not be made unless they empirically improve the goodness of the method at hand, above the mathematically more simple version of the sequential kernel, or sequentialization in Remark **??**.

6. The string, alignment and ANOVA kernel seen via sequentializations

In this section we show how the sequential kernel is closely related to the existing kernels for sequential data:

(a) String kernels [17, 19] may be seen as a special case of the sequential kernel.

(b) The global alignment kernel [5, 6] can be obtained from a special case of the sequential kernel by deleting terms that destroy positive definiteness.

(c) The ANOVA kernel arises by considering only the symmetric part of tensors in $\mathrm{T}(\mathcal{H})$.

(d) The sequential kernel may be understood in the general framework of the relation-convolution kernel [14].

The link will be established to one of the more general variants of sequentialization presented in Remark 5.10, of form

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (\boldsymbol{\sigma}, \boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})], \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j})}} \kappa(\boldsymbol{i}, \boldsymbol{j}) \cdot \prod_{r=1}^{\ell(\boldsymbol{i})} \mathrm{k}(\boldsymbol{\sigma}[i_r], \boldsymbol{\tau}[j_r]),$$

where $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $\kappa : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{R}$ are suitably chosen.

6.1. The string kernel

The string kernel is a kernel for strings in a fixed alphabet Σ, say $\Sigma = \{\mathrm{A}, \mathrm{B}\}$. A number of variants exist; we will consider the original definition of the string kernel (Definition 1, page 423 of [19]). Modifications may be treated similarly to the below.

Definition 6.1. *Fix a finite alphabet Σ. The string kernel on Σ is*

$$K_\Sigma : \Sigma^+ \times \Sigma^+ \to \mathbb{R} \quad (\boldsymbol{\sigma}, \boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})]}} \lambda^{d(\boldsymbol{i}) + d(\boldsymbol{j})} \cdot \mathbb{1}(\boldsymbol{\sigma}[\boldsymbol{i}] = \boldsymbol{\tau}[\boldsymbol{j}]) = \sum_{u \in \Sigma^+} \sum_{\boldsymbol{i} : u = \boldsymbol{\sigma}[\boldsymbol{i}]} \sum_{\boldsymbol{j} : u = \boldsymbol{\tau}[\boldsymbol{j}]} \lambda^{d(\boldsymbol{i}) + d(\boldsymbol{j})},$$

where $\lambda \in \mathbb{R}^+$ is a parameter, $d(\boldsymbol{i}) := \boldsymbol{i}[\ell(\boldsymbol{i})] - \boldsymbol{i}[1] + 1$ is the distance of the last and first symbol in the sub-string given by $\boldsymbol{i}$, and $\mathbb{1}(\boldsymbol{\sigma}[\boldsymbol{i}] = \boldsymbol{\tau}[\boldsymbol{j}])$ is the indicator function of the event whether $\boldsymbol{\sigma}[\boldsymbol{i}]$ and $\boldsymbol{\tau}[\boldsymbol{j}]$ agree, i.e., one if $\boldsymbol{\sigma}[\boldsymbol{i}] = \boldsymbol{\tau}[\boldsymbol{j}]$ and zero otherwise.

For $\lambda = 1$, one has

$$K_\Sigma(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})]}} \mathbb{1}(\boldsymbol{\sigma}[\boldsymbol{i}] = \boldsymbol{\tau}[\boldsymbol{j}]) = \sum_{\substack{\boldsymbol{\sigma}' \sqsubset \boldsymbol{\sigma} \\ \boldsymbol{\tau}' \sqsubset \boldsymbol{\tau} \\ \ell(\boldsymbol{\sigma}') = \ell(\boldsymbol{\tau}')}} \prod_{r=1}^{\ell(\boldsymbol{\sigma}')} \mathrm{k}(\boldsymbol{\sigma}'[r], \boldsymbol{\tau}'[r])$$

for the choice of primary kernel $\mathrm{k}(a, b) = \mathbb{1}(a = b)$ for symbols $a, b \in \Sigma$. Comparing with Proposition 5.5, this canonically identifies the string kernel with parameter $\lambda = 1$ with the sequentialization of the Euclidean scalar product $\mathrm{k}(\cdot, \cdot) = \langle \cdot, \cdot \rangle$ on $\mathcal{X} = \mathbb{R}^{|\Sigma|}$, via identifying a string $\boldsymbol{a} \in \Sigma^L$ with the sequence

$$(0, e_{\boldsymbol{a}[1]}, e_{\boldsymbol{a}[1]} + e_{\boldsymbol{a}[2]}, \ldots, \sum_{r=1}^{L} e_{\boldsymbol{a}[r]}) \in \mathcal{X}^{L+1}.$$

We state the result for arbitrary λ, where one can also express the string kernel as a sequentialization:

Proposition 6.2. *Consider the string kernel K_Σ on an alphabet Σ. Then,*

$$K_\Sigma(\boldsymbol{\sigma}, \boldsymbol{\tau}) = \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})]}} \lambda^{d(\boldsymbol{i}) + d(\boldsymbol{j})} \cdot \mathbb{1}(\boldsymbol{\sigma}[\boldsymbol{i}] = \boldsymbol{\tau}[\boldsymbol{j}]) =: \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j})}} \kappa(\boldsymbol{i}, \boldsymbol{j}) \cdot \prod_{r=1}^{\ell(\boldsymbol{i})} \mathrm{k}(\boldsymbol{\sigma}[i_r], \boldsymbol{\tau}[j_r]$$

with the positive definite kernels $\mathrm{k}(a, b) = \mathbb{1}(a = b)$ and $\kappa(\boldsymbol{i}, \boldsymbol{j}) = \lambda^{d(\boldsymbol{i}) + d(\boldsymbol{j})}$.

Proof. The equality follows from substituting definitions. It remains to show that k,κ are positive definite. For κ, note that we have a scalar product representation $\kappa(\boldsymbol{i},\boldsymbol{j}) = \langle \lambda^{d(\boldsymbol{i})}, \lambda^{d(\boldsymbol{j})}\rangle_{\mathbb{R}}$ (over the real numbers), therefore κ is positive definite. Positive definiteness of k follows similarly from the scalar product representation $\mathrm{k}(a,b) = \sum_{c\in\Sigma^+} \mathbb{1}(a=c)\cdot\mathbb{1}(b=c)$. $\square$

Remark 6.3. *The above shows that string kernel arises a sequentialization as introduced in Section 5. However, it is interesting to note that it is not in exact agreement with the continuous signature kernel $\langle \mathrm{S}(x), \mathrm{S}(y)\rangle$ when we associate with a string $\boldsymbol{a}$ a path*

$$x \in \mathrm{BV}([0,L],\mathbb{R}^{|\Sigma|}) \quad x(t) = \{t\}\cdot e_{\boldsymbol{a}[\lfloor t\rfloor]} + \sum_{r=1}^{\lfloor t\rfloor} e_{\boldsymbol{a}[r]},$$

where as usual $\lfloor t\rfloor$ is the floor function of t, and $\{t\}$ is the fractional part of t (not the set containing t as one element). In fact, one can show that the string kernel cannot be expressed as a signature kernel evaluated at a suitable continuous path. This difference is due to the fact that our sequentialization kernel/discretization is on the level of the tensor algebra and not on the level of paths, see Remark 5.2.

While κ cannot be pulled into the product, the string kernel nevertheless admits a sum-formula presentation similar to Proposition 5.5 (iii), namely

$$K_\Sigma(\boldsymbol{\sigma},\boldsymbol{\tau}) = \sum_{\substack{i_1\geq 1\\ j_1\geq 1}} \lambda^{2-i_1-j_1}\,\mathrm{k}(\boldsymbol{\sigma}[i_1],\boldsymbol{\tau}[j_1]) \sum_{\substack{i_2\gtrsim i_1\\ j_2\gtrsim j_1}} \mathrm{k}(\boldsymbol{\sigma}[i_2],\boldsymbol{\tau}[j_2]) \cdots \sum_{\substack{i_L\gtrsim i_{L-1}\\ j_L\gtrsim j_{L-1}}} \lambda^{i_L+j_L}\,\mathrm{k}(\boldsymbol{\sigma}[i_L],\boldsymbol{\tau}[j_L]),$$

where $L = \min(\ell(\boldsymbol{\sigma}),\ell(\boldsymbol{\tau}))$. Note that each sum runs over a double index, and λ-s occur only next to the first and last sum. The usual convention that a sum running over an empty index set evaluates to zero applies. The sum-formula is well-known and a main tool in efficiently evaluating string kernels, see for example the section "efficient computation of SSK", page 425 of [19].

Gappy and other string kernel variants such as in [17] may be obtained from the truncated sequential kernel and other, suitable choices of κ.

Remark 6.4. *The interpretation as a sequential kernels directly highlights an alternative interpretation, and a natural generalization of the string kernel: suppose each character in the string was not an exact character, but a weighted sum $\alpha A + \beta B$; where for example the character $1A + 0B$ is the same as A, $0A + 1B$ is the same as B, and $\frac{1}{2}A + \frac{1}{2}B$ is the same as half-A-half-B. Such a scenario could practical sense if the exact meaning of a character is not entirely known, for example when the string was obtained through character recognition, for example where the letter l (lower-case-L) and the number 1 are often confounded. It can be observed that this situation can be coped with by mapping say $\frac{1}{2}A + \frac{1}{2}B$ to the vector $\frac{1}{2}e_A + \frac{1}{2}e_B$, with the sequential kernel left unchanged.*

6.2. The global alignment kernel

The global alignment kernel one of the most used kernels for sequences. We recapitulate its definition in modern terminology (Section 2.2 of [5]).

Definition 6.5. *A 2-dimensional integer sequence $\boldsymbol{s} \in \left(\mathbb{N}^2\right)^+$ is called an alignment if $\nabla \boldsymbol{s}[i] \in \{(0,1),(1,0),(1,1)\}$ for all $i \in [\ell(\boldsymbol{s})-1]$. The set of such alignments will be denoted by $\mathcal{A}$; that is, we write $\boldsymbol{s}\in\mathcal{A}$ if $\boldsymbol{s}$ is an alignment.*

Definition 6.6. *Fix an arbitrary set $\mathcal{X}$, and a primary kernel $\mathrm{k}:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$. The global alignment kernel is defined as*

$$K_{GA}:\mathcal{X}^+\times\mathcal{X}^+\to\mathbb{R} \quad (\boldsymbol{\sigma},\boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell(\boldsymbol{\sigma})]\\ \boldsymbol{j}\sqsubseteq[\ell(\boldsymbol{\tau})]\\ \ell(\boldsymbol{i})=\ell(\boldsymbol{j})\leq\ell(\boldsymbol{\sigma})+\ell(\boldsymbol{\tau})\\ (\boldsymbol{i},\boldsymbol{j})\in\mathcal{A}}} \prod_{r=1}^{\ell(I)} \mathrm{k}(\boldsymbol{\sigma}[i_r],\boldsymbol{\tau}[j_r]).$$

In its native form, the global alignment kernel cannot be written as a sequential kernel. A simple proof for this is that the sequential kernels are all positive definite, while the global alignment kernel need not be (see [5]). While one can write

$$K_{\mathrm{GA}}:\mathcal{X}^+\times\mathcal{X}^+\to\mathbb{R}\quad(\boldsymbol{\sigma},\boldsymbol{\tau})\mapsto\sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell(\boldsymbol{\sigma})]\\ \boldsymbol{j}\sqsubseteq[\ell(\boldsymbol{\tau})]\\ \ell(\boldsymbol{i})=\ell(\boldsymbol{j})}}\kappa(\boldsymbol{i},\boldsymbol{j})\cdot\prod_{r=1}^{\ell(\boldsymbol{i})}\mathrm{k}(\boldsymbol{\sigma}[i_r],\boldsymbol{\tau}[j_r])$$

with $\kappa(\boldsymbol{i},\boldsymbol{j})=\mathbb{1}\left((\boldsymbol{i},\boldsymbol{j})\in\mathcal{A}\right)$, this is not a sequential kernel since κ is not positive definite.

However, a simple modification turns the global alignment kernel into a sequential (and thus positive definite) kernel:

Definition 6.7. *A 1-dimensional integer sequence $\boldsymbol{\sigma}\in\left(\mathbb{N}^1\right)^+$ is called a half-alignment if $\nabla\boldsymbol{\sigma}[i]\in\{0,1\}$ for all $i\in[\ell(\boldsymbol{\sigma})-1]$. The set of such half-alignments will be denoted by $\frac{1}{2}\mathcal{A}$; that is, we write $\boldsymbol{\sigma}\in\frac{1}{2}\mathcal{A}$ if $\boldsymbol{\sigma}$ is a half-alignment.*

With this, we can formulate a slightly modified global alignment kernel:

$$K_{\mathrm{G}\frac{1}{2}A}:\mathcal{X}^+\times\mathcal{X}^+\to\mathbb{R}\quad(\boldsymbol{\sigma},\boldsymbol{\tau})\mapsto\sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell(\boldsymbol{\sigma})]\\ \boldsymbol{j}\sqsubseteq[\ell(\boldsymbol{\tau})]\\ \ell(\boldsymbol{i})=\ell(\boldsymbol{j})\le\ell(\boldsymbol{\sigma})+\ell(\boldsymbol{\tau})\\ \boldsymbol{i},\boldsymbol{j}\in\frac{1}{2}\mathcal{A}}}\prod_{r=1}^{\ell(\boldsymbol{i})}\mathrm{k}(\boldsymbol{\sigma}[i_r],\boldsymbol{\tau}[j_r])$$

With a simple re-formulation, one obtains

$$K_{\mathrm{G}\frac{1}{2}A}=\sum_{\substack{\boldsymbol{i}\sqsubseteq[\ell(\boldsymbol{\sigma})]\\ \boldsymbol{j}\sqsubseteq[\ell(\boldsymbol{\tau})]\\ \ell(\boldsymbol{i})=\ell(\boldsymbol{j})\le\ell(\boldsymbol{\sigma})+\ell(\boldsymbol{\tau})}}\kappa'(\boldsymbol{i},\boldsymbol{j})\cdot\prod_{r=1}^{\ell(\boldsymbol{i})}\mathrm{k}(\boldsymbol{\sigma}[i_r],\boldsymbol{\tau}[j_r]),$$

where $\kappa'(\boldsymbol{i},\boldsymbol{j})=\mathbb{1}\left(\boldsymbol{i}\in\frac{1}{2}\mathcal{A}\right)\cdot\mathbb{1}\left(\boldsymbol{j}\in\frac{1}{2}\mathcal{A}\right)$. Note that κ' is positive definite, since it is explicitly given as a scalar product of features in $\mathbb{R}$, thus $K_{\mathrm{G}\frac{1}{2}A}$ is positive definite as a sequential kernel.

The terms missing in K_{GA}, when compared to $K_{\mathrm{G}\frac{1}{2}A}$ are exactly those arising from sequence pairs $(\boldsymbol{i},\boldsymbol{j})\in\left(\mathbb{N}^2\right)^+$ in which there is an increment $(0,0)$. In view of the discussion in Section 3.2 of [5], these missing terms are exactly the locus of non-transitivity in the sense of [34].

One can now follow the authors [5] and heuristically continue studying sufficient conditions under which the original global alignment kernel is positive definite, keeping the missing terms out. However, we would argue, especially in the view of the violated transitivity condition, that it may be more natural to add the missing terms back, unless there is a clear empirical reason in favour of leaving them out. Particularly, we would further argue that there is no first-principles reason to leave the terms out, since the definition of an alignment has been heuristic and somewhat arbitrary to start with.

6.3. The relation-convolution kernel

Finally, we would like to point out that the sequential kernels are closely related to the relation-convolution kernels in the sense of Haussler [14], Section 2.2. We cite it in a slightly less general form than originally defined:

Definition 6.8. *Fix arbitrary sets $\mathcal{Y},\mathcal{Z}$. Further, fix a relation $R\subseteq\mathcal{Y}\times\mathcal{Z}$ and a kernel $\mathrm{k}_{\mathcal{Y}}:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}$. A relation-convolution kernel is a kernel of the form*

$$K_{RC}:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}\quad(x,y)\mapsto\sum_{\substack{r\in R^{-1}(x)\\ s\in R^{-1}(y)}}\mathrm{k}_{\mathcal{Y}}(r,s),$$

where we have written $R^{-1}(x) := \{y : (x,y) \in R\}$.

In the presented form, the signature kernel

$$\mathrm{k}^+ : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}; \quad (\boldsymbol{\sigma}, \boldsymbol{\tau}) \mapsto \sum_{\substack{\boldsymbol{i} \sqsubset [\ell(\boldsymbol{\sigma})] \\ \boldsymbol{j} \sqsubset [\ell(\boldsymbol{\tau})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j})}} \kappa(\boldsymbol{i}, \boldsymbol{j}) \cdot \prod_{r=1}^{\ell(\boldsymbol{i})} \mathrm{k}(\boldsymbol{\sigma}[i_r], \boldsymbol{\tau}[j_r]),$$

is a special case of Haussler's relation-convolution kernel, with the relation being ordered sub-sequence relation. Note though the main discrepancy which is that the native sequential kernel is evaluated at first differences, so a relation-convolution kernel is only obtained after the summation step described in Remark 5.10 (i).

It is also interesting to note that the sub-sequence relation is the same from which the ANOVA kernel is obtained, see Section 2.4 of [14]. However, for the ANOVA kernel, the primary kernel is restricted to be zero for unequal sequence elements.

7. Higher order sequentialization and noisy observations

We have seen in Section 5 that the sequentialisation k^+ of a kernel k converges to the inner product of signature features, that is

$$\mathrm{k}^+(\sigma(\boldsymbol{s}), \tau(\boldsymbol{t})) \to \mathrm{k}^\oplus(\sigma, \tau) = \langle \mathrm{S}\left(\phi(\sigma)\right), \mathrm{S}\left(\phi(\tau)\right)\rangle_{\mathrm{T}(\mathcal{H})} \text{ as } \mathrm{mesh}(\boldsymbol{s}) \vee \mathrm{mesh}(\boldsymbol{t}) \to 0. \tag{7.1}$$

This requires (see Assumption 4.6) that $\phi(\sigma), \phi(\tau) \in \mathrm{BV}(\mathcal{H})$ so that the signature features $\mathrm{S}\left(\phi(\sigma)\right), \mathrm{S}\left(\phi(\tau)\right)$ are well defined via Riemann–Stieljtes integration. However, a common situation is that observations are perturbed by noise in which case the bounded variation assumption is typically not fulfilled. An insight of rough path theory and stochastic analysis is that despite the breakdown of classic integration, it is for large classes of paths possible to define a map that replaces the signature $\sigma \mapsto \mathrm{S}\left(\phi(\sigma)\right)$ or in our learning context: becomes a feature map for σ.

The price is that we need to replace $\mathfrak{S}$ by a *higher order approximation* $\mathfrak{S}_D$ respectively $\langle \mathfrak{S} \circ \phi, \mathfrak{S} \circ \phi \rangle$ by $\langle \mathfrak{S}_D \circ \phi, \mathfrak{S}_D \circ \phi \rangle$, here $D \geq 1$ will denote the order of approximation and has to be choosen higher the more irregular the underlying paths are. A thorough discussion for general noise (truly "rough" paths) requires some background in stochastic analysis and rough path theory and is beyond the scope of this article and we refer to [8, 20]. The point we want to make in this section, is that with a few modifications, *the methodology of the previous chapter extends to sequences that are sampled from unbounded variation paths*.

7.1. Brownian noise: order 2 approximations

Below we demonstrate the needed adapations for multivariate white noise/Brownian motion; moreover, we focus on sequentialization of the trivial kernel $\mathrm{k} = \langle \cdot, \cdot \rangle_{\mathbb{R}^d}$, that is the primary feature map $\phi(x) = x$. In this case, we deal with (semi)martingale paths and an approximation of degree $D = 2$ is needed.

Proposition 7.1. *Let μ be the Wiener measure on $C([0,1]), \mathbb{R}^d)$, that is $x \sim \mu$ is a d-dimensional Brownian motion. Let $\boldsymbol{t}_n \in \Delta$ be dyadics, $\boldsymbol{t}_n[i] := i2^{-n}$, and define $(x_n) \in \mathrm{BV}([0,1], \mathbb{R}^d)$ as the piecewise linear interpolation on $\boldsymbol{t}_n$,*

$$x_n(t) := x(\boldsymbol{t}_n)[i] + (t - \boldsymbol{t}_n[i]) \nabla x(\boldsymbol{t}_n)[i] \text{ for } t \in \left[\boldsymbol{t}_n[i], \boldsymbol{t}_n[i+1]\right). \tag{7.2}$$

For any $p > 2$ and any multindex $(i_1, i_2) \in \{1, \dots d\}^2$ it holds for μ-a.e. x that

1. $\mathrm{V}(x) = \infty$,

2. $\lim_{n\to\infty} \mathrm{S}_{(i_1,i_2)}(x_n)$ *exists in* $\mathrm{T}(\mathcal{H})$ *and equals the stochastic Ito–Stratonovich integral*

$$\int_{\Delta_2([0,1])} \circ \mathrm{d}x_{i_1} \circ \mathrm{d}x_{i_2}, \tag{7.3}$$

3. $\mathfrak{S}_1(x(\boldsymbol{t}_n))_{(i_1,i_2)}$ *does not converge to (7.3) as* $n\to\infty$ *if* $i_1 = i_2$.

However, if we denote

$$\mathfrak{S}_{(2)} : \mathcal{H}^+ \to \mathrm{T}(\mathcal{H}), \quad \mathfrak{S}_{(2)}(\boldsymbol{x}) = \prod_{i=1}^{\ell(\nabla x)} \left(1 + \nabla \boldsymbol{x}[i] + \frac{(\nabla \boldsymbol{x}[i])^{\otimes 2}}{2!}\right) \tag{7.4}$$

then the (i_1,i_2) *coordinate of* $\mathfrak{S}_{(2)}\left(x(\boldsymbol{t}_n)\right)$ *converges to (7.3) as* $n\to\infty$ *for* μ*-a.e.* x.

Proof. Point (1) can be found in any textbook on stochastic analysis, e.g. [28]; similarly, point (2) is classic, e.g. [8][Chapter 13]. Also point (3) and the convergence of $\mathfrak{S}_x(x_n)$ follows by a simple calculation:

$$\begin{aligned}
\mathfrak{S}_2(x_n) &= \prod_{i=1}^{\ell(\boldsymbol{t}_n)-1} \left(1 + \nabla x(\boldsymbol{t}_n)[i] + \frac{(\nabla x(\boldsymbol{t}_n)[i])^{\otimes 2}}{2}\right) \\
&= 1 + \sum_{i\in[\ell(\boldsymbol{t}_n)-1]} \nabla x(\boldsymbol{t}_n)[i] + \sum_{i,j\in[\ell(\boldsymbol{t}_n)-1], i<j} \nabla x(\boldsymbol{t}_n)[i] \otimes \nabla x(\boldsymbol{t}_n)[j] + \frac{1}{2} \sum_{i\in[\ell(\boldsymbol{t}_n)-1]} \nabla x(\boldsymbol{t}_n)[i]^{\otimes 2}.
\end{aligned}$$

Now the first sum equals, $x(\boldsymbol{t}_n[\ell(\boldsymbol{t}_n)]) = \int_{\Delta_1(0,1)} \mathrm{d}x$, the second equals

$$\sum_{j\in[\ell(\boldsymbol{t}_n)-1]} \sum_{i\in[j-1]} \nabla \boldsymbol{x}[i] \otimes \nabla \boldsymbol{x}[j] = \sum_{j\in[\ell(\boldsymbol{t}_n)-1]} x(\boldsymbol{t}_n[i]) \otimes \nabla x(\boldsymbol{t}_n[i]) \tag{7.5}$$

and a classic convergence results in stochastic calculus,[28], shows that

$$\sum_{j\in[\ell(\boldsymbol{t}_n)-1]} x(\boldsymbol{t}_n[i]) \otimes \nabla x(\boldsymbol{t}_n[i]) + \frac{1}{2} \sum_{i\in[\ell(\boldsymbol{t}_n)-1]} \nabla x(\boldsymbol{t}_n[i])^{\otimes 2} \tag{7.6}$$

converges to the Stratonovich integral $\int \circ \mathrm{d}x \circ \mathrm{d}x$. Now point (3) follows by recalling that Brownian motion has non-vanishing quadratic variation, that is $\sum_{i\in[\ell(\boldsymbol{t}_n)-1]} \nabla x(\boldsymbol{t}_n[i]))^{\otimes 2}$ does not converge to 0 as $n\to\infty$. □

Corollary 7.2. *Let* μ *be the Wiener measure on* $C([0,1]),\mathbb{R}^d)$. *Denote the* $\mathfrak{S}_{(2),2}$ *the projection of* $\mathfrak{S}_{(2)}$ *to* $\bigoplus_{m=0}^{2}(\mathbb{R}^d)^{\otimes m}$. *If* $x,y \sim \mu$ *independently, then with probability one*

$$\langle \mathfrak{S}_{(2),2}(x(\boldsymbol{s}_m)), \mathfrak{S}_{(2),2}(y(\boldsymbol{t}_n))\rangle_{\mathrm{T}(\mathcal{H})} \to \langle X, Y\rangle_{\mathrm{T}(\mathcal{H})} \text{ as } m,n\to\infty \tag{7.7}$$

where $X := 1 + \int_{\Delta_1} \circ \mathrm{d}x + \int_{\Delta_2} (\circ \mathrm{d}x)^{\otimes 2}$ *and* $Y := 1 + \int_{\Delta_1} \circ \mathrm{d}y + \int_{\Delta_2} (\circ \mathrm{d}y)^{\otimes 2}$ *are given by Ito–Stratonovich integrals and* $\boldsymbol{s}_m$, $\boldsymbol{t}_n$ *are dyadic partitions,* $\boldsymbol{s}[i] = i2^{-m}$, $\boldsymbol{t}[i] = i2^{-n}$.

Remark 7.3. *Essentially the same result holds for other partitions than dyadics that vanish quickly enough and for continuous semimartingales, that is bounded variation paths with additive noise of the same path regularity as Brownian motion (see [8]). Similarly, one can show that* $\mathfrak{S}(x(\boldsymbol{t}_n))$ *converges for higher iterated integrals though the calculation gets a bit cumbersome so we do not address this. Useful tools to study such convergence questions are the notions of* multiplicative functionals *and* extension theorem*; we refer to [22].*

7.2. Higher order approximations

In the Brownian (and semimartingale) discussed above, the Riemann–Stieljtes integral was replaced by stochastic Ito–Stratonovich integral, thus providing a map from (semimartingale) paths to $\mathrm{T}(\mathcal{H})$. A general strategy to construct a function that maps a path x to $\mathrm{T}(\mathcal{H})$ and behaves like iterated integrals is to find a sequence of bounded variation paths $(x_n) \subset \mathrm{BV}([0,1],\mathcal{H})$ such that the $\mathrm{T}(\mathcal{H})$-valued paths $t \mapsto \sum_m \int_{\Delta([0,t])}(\mathrm{d}x_n)$ converges in an apropiate sense to a $\mathrm{T}(\mathcal{H})$-valued path denoted $t \mapsto X(t)$. This is the notion of a *geometric rough path* and allows to study classes of much "rougher" paths; loosely speaking: *if we deal with a geometric p-rough path, then we have to consider an approximation $\mathfrak{S}_{(D)}$ of order at least* $D = \lfloor p \rfloor$; thus the rougher the path, the higher the order of signature approximation.

Example 7.4. *As pointed out above, the details of how to map a trajectory to an element in* $\mathrm{T}(\mathcal{H})$ *vary, exploit probabilistic structure and we refer [8, 20] for details. Here we just mention that such constructions are well-known for*

- 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).*

Again, a rigorous treatment requires knowledge of geometric p-rough paths and is beyond the aim of this paper. However, we still give the general definition of a an order D approximation and the associated signature and sequentialized kernel that are needed to treat the paths in Example 7.4.

Definition 7.5. *Let $D \in \mathbb{N}$. Define*

$$\mathfrak{S}_{(D)} : \mathcal{H}^+ \to \mathrm{T}(\mathcal{H}), \quad \mathfrak{S}_{(D)}(\boldsymbol{x}) = \prod_{i=1}^{\ell(\nabla \boldsymbol{x})} \sum_{m=0}^{D} \frac{(\nabla \boldsymbol{x}[i])^{\otimes m}}{m!}$$

and denote with $\mathfrak{S}_{(D),M}$ the projection of $\mathfrak{S}_{(D)}$ to $\bigoplus_{m=0}^{M} \mathcal{H}^{\otimes m}$. Let $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Define the sequentialization of k *of order D and truncated at M as*

$$\mathrm{k}^+_{(D),M}(\boldsymbol{\sigma},\boldsymbol{\tau}) : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}, \quad \mathrm{k}^+_{(D),M}(\boldsymbol{\sigma},\boldsymbol{\tau}) = \langle \mathfrak{S}_{(D),M}\left(\phi(\boldsymbol{\sigma})\right), \mathfrak{S}_{(D),M}\left(\phi(\boldsymbol{\tau})\right)\rangle_{\mathrm{T}(\mathcal{H})}.$$

Remark 7.6. *The sequnentialization k^+_M of a kernel* k *from Section 5.1 arises as special case of above, general definition: $\mathrm{k}^+_M = \mathrm{k}^+_{(D),M}$ for $D = 1$, $M \in \mathbb{N}$.*

In analogy to the order $D = 1$ approximations discussed in Section 5, the central mathematical identity is now

$$\prod_{i=1}^{\ell(\nabla \boldsymbol{x})} \sum_{m=0}^{D} \frac{(\nabla \boldsymbol{x}[i])^{\otimes m}}{m!} = \sum_{\boldsymbol{i} \sqsubseteq_D [\ell(\nabla \boldsymbol{x})]} \frac{1}{\boldsymbol{i}!} \prod_{r=1}^{\ell(\boldsymbol{i})} \nabla \boldsymbol{x}[i_r] \approx \sum_{m=0}^{\infty} \int_{\Delta^m([0,1])} \mathrm{d}x^{\otimes m}$$

where $\boldsymbol{x} = x(\boldsymbol{t})$ is a suitable discretization of the (unbounded variation!) path $x \in C([0,1],\mathcal{H})$ with $\boldsymbol{t} \in \Delta([0,1])$. Note the appearance of the $\boldsymbol{i}!$ term in the first identity together with the restriction $\boldsymbol{i} \sqsubseteq_D [\ell(\nabla \boldsymbol{x})]$ which makes a recursion formula for the inner product more complex. However, a variation of the recursive Horner type formula still holds for $\mathrm{k}^+_{(D),M}$ and we give an efficient algorithm for $\mathrm{k}^+_{(D),M}$ for general[7] $D, M \in \mathbb{N}$ in Section 8 below. Since it is a multi-way recursion which is better formulated in terms of a dynamic programming algorithm, we defer its formulation to the following Section 8. It relies on the formula below for the discretized higher order signature kernel:

Proposition 7.7. *For $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}^+$, $D, M \in \mathbb{N}$,*

(a) $\mathfrak{S}_{(D)}(\boldsymbol{x}) = \sum_{\boldsymbol{i} \sqsubseteq_D [\ell(\nabla \boldsymbol{x})]} \frac{1}{\boldsymbol{i}!} \prod_{r=1}^{\ell(\boldsymbol{i})} \nabla \boldsymbol{x}[i_r]$,

(b) $\langle \mathfrak{S}_{(D),M}(\boldsymbol{x}), \mathfrak{S}_{(D),M}(\boldsymbol{y}) \rangle_{\mathrm{T}(\mathcal{H})} = \sum_{\substack{\boldsymbol{i},\boldsymbol{j} \in \Delta(\mathbb{N}) \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j}) \leq M \\ \#\boldsymbol{i}, \#\boldsymbol{j} \leq D}} \frac{1}{\boldsymbol{i}!\boldsymbol{j}!} \prod_{r=1}^{\ell(\boldsymbol{i})} \langle \nabla \boldsymbol{x}[i_r], \nabla \boldsymbol{y}[j_r] \rangle_{\mathcal{H}}$.

Proof. Writing out the definition of $\mathfrak{S}_{(D)}(\boldsymbol{x})$ yields

$$\mathfrak{S}_{(D)}(\boldsymbol{x}) = \prod_{r=1}^{\ell(\nabla \boldsymbol{x})} \sum_{d=0}^{D} \frac{1}{d!} (\nabla \boldsymbol{x}[r])^{\otimes d}.$$

Application of the (non-commutative) associative law yields

$$\prod_{r=1}^{\ell(\nabla \boldsymbol{x})} \sum_{d=0}^{D} \frac{1}{d!} (\nabla \boldsymbol{x}[r])^{\otimes d} = \sum_{\boldsymbol{i} \sqsubseteq_D [\ell(\nabla \boldsymbol{x})]} \frac{1}{\boldsymbol{i}!} \prod_{r=1}^{\ell(\boldsymbol{i})} \nabla \boldsymbol{x}[i_r].$$

The analogous expression for $\mathfrak{S}_{(D)}(\boldsymbol{y})$ and taking the scalar product while noting

$$\langle \prod_{r=1}^{\ell(\boldsymbol{i})} \nabla \boldsymbol{x}[i_r], \prod_{r'=1}^{\ell(\boldsymbol{j})} \nabla \boldsymbol{y}[j_{r'}] \rangle_{\mathrm{T}(\mathcal{H})} = \delta_{\ell(\boldsymbol{i}),\ell(\boldsymbol{j})} \cdot \prod_{r=1}^{\ell(\boldsymbol{i})} \langle \nabla \boldsymbol{x}[i_r], \nabla \boldsymbol{y}[j_r] \rangle_{\mathcal{H}}.$$

Truncating at tensor degree M yields the claim. □

8. Efficient computation of sequentialized kernels

Naive evaluation of the signature kernels K_M^+ and sequentialized kernels k_M^+ (or more generally $\mathrm{k}_{(D),M}^+$) incurs a cost exponential in the length of the input or the degree. Inspired by dynamic programming, we present in this section a number of algorithms for signature and sequential kernels whose time and memory requirements are polynomial in the length of the sequence.

Table 1 presents an overview on the different algorithms presented: given N sequences, $\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_N \in \mathcal{X}^+$, each of length less than L, that is $\ell(\boldsymbol{\sigma}_i) \leq L$ for $i = 1, \ldots, N$, Table 1 shows the computational complexity and storage requirements for calculating the Gram-matrix $\mathrm{k}_M^+ \left(\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_j\right)_{i,j=1}^N$ of the sequentialization k_M^+ of a kernel k on $\mathcal{X}$. The simplest variant involving dynamic programming, Algorithm 3, allows to evaluate the sequential kernel in a number of elementary computations that is linear in the length of either sequence (thus quadratic for two sequences of equal length). Further combining the strategy with low-rank ideas allows to compute a full sequential kernel matrix that is both linear in the maximum length of sequence and the number of sequences, culminating in Algorithm 6.

The higher order sequential kernel $\mathrm{k}_{(D),M}^+$ will not be discussed to the same degree of detail. It is demonstrated in Algorithm 4 how the simple dynamic programming Algorithm 3 changes when the approximation is carried out to a higher order D. All dynamic programming algorithms listed in Table 1 may be modified in the same way while incurring an additional factor D in computation and storage requirements.

[7]Without loss of generality, it is sufficient to consider $D \leq M$ since for $D > M$ they only difference occurs at the level of more than M times iterated integrals and we already cut off at degree M).

method	algorithm	complexity	storage
naive evaluation	Proposition 5.5	$O(N^2 \cdot L^{2M})$	$O(1)$
dynamic programming	Algorithm 3	$O(N^2 \cdot L^2 \cdot M)$	$O(L^2)$
DP & LR, per-element	Algorithm 5	$O(N^2 \cdot L \cdot \rho \cdot M)$	$O(L \cdot \rho)$
DP & LR, per-sequence	Algorithm 3	$O((N+\rho) \cdot L^2 \cdot \rho^2 \cdot M)$	$O(N \cdot L^2)$
DP & LR, simultaneous	Algorithm 6	$O(N \cdot L \cdot \rho \cdot M)$	$O(N \cdot L \cdot \rho)$

Table 1: Overview over presented algorithms to compute the sequential kernel k^+_M, their computational cost and storage cost when evaluating an $(N \times N)$ kernel matrix between sequences of length at most L. Methods: DP = dynamic programming, LR = low-rank. In the low-rank methods, ρ is a meta-parameter which also controls accuracy of approximation and prediction, not necessarily in the same way for the different algorithms.

Remark 8.1. *Let $\sigma, \tau \in \mathrm{BV}(\mathbb{R}^d)$, $\boldsymbol{s}, \boldsymbol{t} \in \Delta$ with $\ell(\boldsymbol{s}), \ell(\boldsymbol{t}) \leq L$. Denote with $\sigma^{\boldsymbol{s}}, \tau^{\boldsymbol{t}} \in \mathrm{BV}(\mathbb{R}^d)$ the bounded variation paths that are the piecewise linear interpolation of points $\sigma(\boldsymbol{s}), \tau(\boldsymbol{t})$. Note that $\mathrm{S}_{\leq M}(\sigma^{\boldsymbol{s}}), \mathrm{S}_{\leq M}(\tau^{\boldsymbol{t}})$ are composed of $O(d^M)$ real numbers which makes a naive evaluation of $\langle \mathrm{S}_{\leq M}(\sigma^{\boldsymbol{s}}), \mathrm{S}_{\leq M}(\tau^{\boldsymbol{t}}) \rangle_{\mathrm{T}(\mathcal{H})}$ infeasible for moderately high d or M. On the other hand, Table 1 applied with $\mathcal{X} = \mathbb{R}^d$, $\mathrm{k} = \langle \cdot, \cdot \rangle_{\mathbb{R}^d}$ and $D = M$, provides efficient methods for calculating this inner product since $\mathrm{k}^+_{(D),M}(\sigma(\boldsymbol{s}), \tau(\boldsymbol{t})) = \langle \mathrm{S}_{\leq M}(\sigma^{\boldsymbol{s}}), \mathrm{S}_{\leq M}(\tau^{\boldsymbol{t}}) \rangle_{\mathrm{T}(\mathcal{H})}$.*

8.1. Dynamic programming for tensors

Before giving algorithms for computing the sequential kernels, we introduce a number of notations and fast algorithms for dynamic programming subroutines which will allow to state the latter algorithms concisely and which are at the basis of fast computations.

Notation 8.2. *We will denote the $(i_1, \ldots, i_k)$-th element of a k-fold array (= degree k tensor) A by $A[i_1, \ldots, i_k]$. Occasionally, for ease of reading, we will use "|" instead of "," as a separator, for example $A[i_1, i_2 | i_3, \ldots, i_k]$ of the indices on the left side of "|" are semantically distinct from those on the right side, in the example to separate the group of indices i_1, i_2 from the group $i_3, \ldots, i_k$. The arrays in the remainder will all contain elements in $\mathbb{R}$, and the indices will always be positive integers, excluding zero.*

Notation 8.3. *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., $f(A)[i_1, \ldots, i_k] = f(A[i_1, \ldots, i_k])$. Similarly, for $f : \mathbb{R}^m \to \mathbb{R}$ and arrays $A_1, \ldots, A_m$, we denote*

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

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 8.4. *We introduce notation for sub-setting, shifting, and 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 $A[:, \ldots, :, i_j, :, \ldots, :]$ for $(k-1)$-fold array of size $(n_1 \times \cdots \times n_{j-1} \times n_{j+1} \times \ldots n_k)$ such that $A[:, \ldots, :, i_j, :, \ldots, :][i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_k] = A[i_1, \ldots, i_k]$. We define in analogy, iteratively, $A[:, \ldots, :, i_j, :, \ldots, :, i_{j'}, :, \ldots, :]$, and so on. Arrays of this type are called slices (of A).*

(ii) *For an integer m, we will write $A[:, \ldots, :, +m, :, \ldots, :]$ for the k-fold array of size $(n_1 \times \cdots \times n_{j-1} \times (n_j + k) \times n_{j+1} \times \ldots n_k)$ such that $A[:, \ldots, :, i_j, :, \ldots, :][i_1, \ldots, i_{j-1}, i_j + k, i_{j+1}, \ldots, i_k] = A[i_1, \ldots, i_k]$, 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 $A[:, \ldots, :, \boxplus, :, \ldots, :]$, where $\boxplus$ is at the j-th position, for the k-fold array of size $(n_1 \times \cdots \times \ldots n_k)$ such that $A[:, \ldots, :, \boxplus, :, \ldots, :][i_1, \ldots, i_k] = \sum_{\kappa=1}^{i_j} A[i_1, \ldots, i_{j-1}, \kappa, i_{j+1}, \ldots, i_k]$. 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* $(n_1\times\dots\times n_{j-1}\times n_{j+1}\times\dots n_k)$ *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 cumulation commute with each other. Therefore expressions such as $A[+1,:|\Sigma,-3]$ *or* $A[j|:,+3,\boxplus]$ *are well-defined, for example. We will also use the notation* $A[\boxplus-m,\dots]$ *and* $A[\boxplus+m,\dots]$ *to indicate the shifted variant of the cumulative sum array.*

Before continuing, we would like to note that 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:     $Q[\kappa] \leftarrow Q[\kappa-1]+A[\kappa]$
4: end for
5: Return $Q$
```

Algorithm 2 Computing the cumulative sum of an array.
Input: A k-fold array A of size $(n_1,\times,\dots,\times n_k)$
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:     Let $Q \leftarrow Q[:,\dots,:,\boxplus,:,\dots,:]$ (at the $\kappa$-th index), where the right side is computed via applying
   algorithm 1 slice-wise.
4: end for
5: Return $Q$
```

8.2. Computing the sequential kernel

We give a fast algorithm to compute the sequential kernel k^+_M, by using the recursive presentation from Proposition 5.5.

Algorithm 3 includes a cut-off degree M which for exact computation can be set to $M=\min(L,L')$ but can be set lower to reduce computational cost when an approximation is good enough.

Note that following our convention on arrays, all multiplications in Algorithm 3 are entry-wise, not matrix multiplications, even though some arrays have the format of compatible matrices.

Correctness of Algorithm 3 is proved in Proposition 5.5. At the end of the algorithm, the array A contains as elements $A[m|i,j]$ the contributions from sub-sequences $\boldsymbol{i}\sqsubseteq[i],\boldsymbol{j}\sqsubseteq[j]$, beginning at i and j, and of total length at most m.

Disregarding the cost of computing K which can vary depending on the exact form of k (but which is, usually, $O(n\ell(\boldsymbol{\sigma})\ell(\boldsymbol{\tau}))$ time and $O(\ell(\boldsymbol{\sigma})\ell(\boldsymbol{\tau}))$ time, with constants that may depend on k), the computational cost of Algorithm 3 is $O(M\ell(\boldsymbol{\sigma})\ell(\boldsymbol{\tau}))$ elementary arithmetic operations (= the number of loop elements) and $O(M\ell(\boldsymbol{\sigma})\ell(\boldsymbol{\tau})$ units of elementary storage. The storage requirement can be reduced to $O(\ell(\boldsymbol{\sigma})\ell(\boldsymbol{\tau}))$ by discarding $A[m-1|:,:]$ from memory after step 7 each time.

Algorithm 3 Evaluation of the sequential kernel k^+_M
Input: Ordered sequences $\boldsymbol{\sigma}, \boldsymbol{\tau} \in \mathcal{X}^+$. A kernel $\mathrm{k} : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$ to sequentialize. A cut-off degree M.
Output: $\mathrm{k}^+_M(\boldsymbol{\sigma}, \boldsymbol{\tau})$, as the sequentialization of k

```
 1: Compute an $(L \times L')$ array $K$ such that $K[i,j] = \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i,j]$.
 2: (or, alternatively, obtain it as additional input to the algorithm)
 3: Initialize an $(M \times L \times L')$-array $A$.
 4: Set $A[1| :,:] \leftarrow K$.
 5: for $m = 2$ to $M$ do
 6:     Compute $Q \leftarrow A[m-1|\boxplus,\boxplus]$.
 7:     Set $A[m| :,:] \leftarrow K \cdot (1 + Q[+1,+1])$
 8: end for
 9: Compute $R \leftarrow 1 + A[M|\Sigma,\Sigma]$
10: Return $R$
```

Note that in each loop over the index L, a matrix Q is pre-computed, to avoid a five-fold loop that would be necessary with the more naive version of line 7,

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

that leads to a blown up computational cost of $O(M\ell(\boldsymbol{\sigma})^2 \ell(\boldsymbol{\tau})^2)$ at the asymptotically insignificant gain of storing one $(\ell(\boldsymbol{\sigma}) \times \ell(\boldsymbol{\tau}))$ matrix less (the matrix Q).

Note that the whole code can be directly translated to the vector of matrix operations commonly available in programming languages such as R, MATLAB, or Python.

8.3. Computing the higher order sequential kernel

Algorithm 4 Evaluation of the higher order sequential kernel $\mathrm{k}^+_{(D),M}$
Input: Ordered sequences $\boldsymbol{\sigma}, \boldsymbol{\tau} \in \mathcal{X}^+$. A kernel $\mathrm{k} : \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$ to sequentialize. A cut-off degree M, an approximation order $D, D \leq M$.
Output: $\mathrm{k}^+_{(D),M}(\boldsymbol{\sigma}, \boldsymbol{\tau})$, as the sequentialization of k

```
 1: Compute an $(L \times L')$ array $K$ such that $K[i,j] = \nabla \mathrm{k}(\boldsymbol{\sigma}, \boldsymbol{\tau})[i,j]$.
 2: (or, alternatively, obtain it as additional input to the algorithm)
 3: Initialize an $(M \times D \times D \times L \times L')$-array $A$, all entries zero.
 4: for $m = 2$ to $M$ do
 5:     $D' \leftarrow \min(D, m-1)$
 6:     $A[m|1,1| :,:] \leftarrow K \cdot (1 + A[m-1|\Sigma,\Sigma|\boxplus+1,\boxplus+1])$
 7:     for $d = 2$ to $D'$ do
 8:         $A[m|d,1| :,:] \leftarrow A[m|d,1| :,:] + \frac{1}{d} \cdot K \cdot A[m-1|d-1,\Sigma| :,\boxplus+1]$.
 9:         $A[m|1,d| :,:] \leftarrow A[m|1,d| :,:] + \frac{1}{d} \cdot K \cdot A[m-1|\Sigma,d-1|\boxplus+1,:]$.
10:         for $d' = 2$ to $D'$ do
11:             $A[m|d,d'| :,:] \leftarrow A[m|d,d'| :,:] + \frac{1}{dd'} \cdot K \cdot A[m-1|d-1,d'-1| :,:]$.
12:         end for
13:     end for
14: end for
15: Compute $R \leftarrow 1 + A[M|\Sigma,\Sigma|\Sigma,\Sigma]$
16: Return $R$
```

All multiplications in Algorithm 4 are entry-wise, not matrix multiplications. At the end of Algorithm 4, the array A contains as elements $A[m|d,d'|i,j]$ the contributions from sub-sequences $\boldsymbol{i} \subseteq [i], \boldsymbol{j} \subseteq [j]$, beginning at i and j, with end-sequences $iii\dots$ of length d and $jjj\dots$ of length d', and of total length at most M. We prove the recursion used in Algorithm 4 and thus the correctness of this statement after introducing some necessary notation:

Notation 8.5. *Let $\boldsymbol{t} = (t_1, \dots, t_M) \in U^M$ be a sequence, for any set U, and some $M \in \mathbb{N}$. We will denote by*

$$d(\boldsymbol{t}) := \max\{m \,:\, t_1 = t_2 = \dots = t_m\}$$

the number of repetitions of the initial symbol.

Proposition 8.6. *Keep the notations of Algorithm 3, let $\phi : \mathcal{X} \to \mathcal{H}$ the feature map associated with the kernel* k*. Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{H}^+$ such that $\boldsymbol{x} = \phi(\boldsymbol{\sigma}), \boldsymbol{y} = \phi(\boldsymbol{\tau})$. Denote by*

$$A[m|d,d'|i,j] := \sum_{\substack{\boldsymbol{x}' \sqsubseteq_D \nabla \boldsymbol{x} \\ \boldsymbol{y}' \sqsubseteq_D \nabla \boldsymbol{y}}} \frac{1}{\boldsymbol{x}'!\boldsymbol{y}'!} \langle \boldsymbol{x}', \boldsymbol{y}' \rangle_{\mathcal{H}^+} = \sum_{\substack{\boldsymbol{i} \sqsubseteq_D [\ell(\boldsymbol{x})] \\ \boldsymbol{j} \sqsubseteq_D [\ell(\boldsymbol{y})] \\ \ell(\boldsymbol{i}) = \ell(\boldsymbol{j}) \le m}} \frac{1}{\boldsymbol{i}!\boldsymbol{j}!} \prod_{\kappa=1}^{\ell(\boldsymbol{i})} \langle (\nabla \boldsymbol{x})[i_\kappa], (\nabla \boldsymbol{y})[j_\kappa] \rangle_{\mathcal{H}}$$

where the sums are additionally restricted in the following way:

$$i = i_1 = i_2 = \dots = i_d \neq i_{d+1} \quad \text{and} \quad j = j_1 = j_2 = \dots = j_{d'} \neq j_{d'+1}. \quad \text{Or, equivalently,}$$
$$\boldsymbol{x}[i] = \boldsymbol{x}'[1] = \boldsymbol{x}'[2] = \dots = \boldsymbol{x}'[d] \neq \boldsymbol{x}'[d+1] \quad \text{and} \quad \boldsymbol{y}[j] = \boldsymbol{y}'[1] = \boldsymbol{y}'[2] = \dots = \boldsymbol{y}'[d] \neq \boldsymbol{y}'[d+1]$$

Then, the following recursion equalities hold:

$$A[m|1,1|i,j] = \langle \boldsymbol{x}[i], \boldsymbol{y}[j] \rangle_{\mathcal{H}} \cdot \left(1 + \sum_{i' \gtrsim i} \sum_{j' \gtrsim j} A[m-1|d-1,d'-1|i',j'] \right),$$
$$A[m|d,1|i,j] = \frac{1}{d} \cdot \langle \boldsymbol{x}[i], \boldsymbol{y}[j] \rangle_{\mathcal{H}} \cdot \sum_{j' \gtrsim j} \sum_{\kappa=1}^{D} A[m-1|d-1,\kappa|i,j'] \quad \text{for } d \ge 2,$$
$$A[m|1,d'|i,j] = \frac{1}{d'} \cdot \langle \boldsymbol{x}[i], \boldsymbol{y}[j] \rangle_{\mathcal{H}} \cdot \sum_{i' \gtrsim i} \sum_{\kappa=1}^{D} A[-1|\kappa,d'-1|i',j] \quad \text{for } d' \ge 2,$$
$$A[m|d,d'|i,j] = \frac{1}{dd'} \cdot \langle \boldsymbol{x}[i], \boldsymbol{y}[j] \rangle_{\mathcal{H}} \cdot A[m-1|d-1,d'-1|i,j] \quad \text{for } d,d' \ge 2.$$

Proof. Note that the sums on the right hand side that define $A[m|d,d'|i,j]$ as a (weighted) sum over elements parameterised by paired index sequences I, J of the same length. Note that by definition, the sum goes over all index sequences such that $\ell(\boldsymbol{i}) = \ell(\boldsymbol{j}) \le M$, $d(\boldsymbol{i}) = d$, and $d(\boldsymbol{j}) = d'$.

With this, the statement follows from comparing the summation in the loop between Line 5 and 13 of Algorithm 4 with the explicit formula in Proposition 7.7. ☐

The computational cost of Algorithm 4 is $O(D^2 M \ell(\boldsymbol{\sigma}) \ell(\boldsymbol{\tau}))$ elementary arithmetic operations (= the number of loop elements) and $O(D^2 \ell(\boldsymbol{\sigma}) \ell(\boldsymbol{\tau}))$ units of elementary storage (when freeing up space for array entries directly after the last time they are read out).

8.4. Large scale strategies

Even though a computational cost of $O(M \ell(\boldsymbol{\tau}) \ell(\boldsymbol{\sigma}))$ for the sequential kernel via Algorithm 3 (for ease of reading, we will not discuss the higher order kernel, most considerations hold in analogy) can be considered efficient in a polynomial time, one has to note that this is the cost of evaluating $\mathrm{k}^+_M(\boldsymbol{\sigma}, \boldsymbol{\tau})$ for a single pair of sequences $\boldsymbol{\sigma}, \boldsymbol{\tau} \in \mathcal{X}^+$. Thus, computation of a symmetric kernel matrix of N sequences, of length at most M, would cost $O(N^2 \cdot L^2 \cdot M)$ elementary arithmetic operations when done by iterating over entries. While this is for moderate sizes of N and M still achievable on contemporary desktop computers, it may become quickly prohibitive when combined with parameter tuning or cross-validation schemes (as later in our experiments). Furthermore, there exist regimes (low signal dimension n, large length L) in which an explicit computation of features plus subsequent inner product, with a complexity of $O(N^2 \cdot L \cdot n^M)$, may be faster due to the linear dependence on L at the cost of an exponential dependence on M.

We present below a number of approaches by which the above-mentioned issues may be addressed. These are somewhat independent and address different parts of the total complexity in different ways, but can be in-principle combined.

8.4.1. Low-rank methods for the sequence-vs-sequence kernel matrix The sequential kernel is a kernel on ordered data, therefore learning algorithms which use the kernel matrix as an input, such as support vector machines, kernel ridge regression, or kernel principal components, are in-principle 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 $O((r+N) \cdot r \cdot L^2 \cdot M)$ elementary operations and $O((r+N) \cdot r \cdot L^2)$ storage, 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).

This alleviates the dependency of the sequential kernel evaluation on N, but not on L; which is not unexpected, since the strategy is completely independent of how the sequential kernel was evaluated. For the same reason, any improvements on the cost of single evaluations will combine with the above improvement un the number of evaluations.

8.4.2. Low-rank methods for the element-vs-element kernel matrix A second kernel matrix is crucial to obtaining a single evaluation of type $\mathrm{k}^+_M(\boldsymbol{\sigma}, \boldsymbol{\tau})$, namely the cross-kernel matrix between the elements $\boldsymbol{\sigma}[i], \boldsymbol{\tau}[j]$ of both sequences which is the object underlying the computations, see Line 2 of Algorithm 3, and Line 2 of Algorithm 4. Summations and multiplications are performed on this cross-kernel matrix until the final result, $\mathrm{k}^+_M(\boldsymbol{\sigma}, \boldsymbol{\tau})$, is obtained and returned.

The low-rank methods of the previous paragraph cannot be applied naively to the cross-kernel matrix. A minor issue is the fact that the cross-kernel matrix is in general non-symmetric, but the above-mentioned strategies (incomplete Cholesky, Nyström, inducing points) translate verbatim to the context of non-symmetric cross-kernel matrices, by replacing the respective symmetric decomposition with the analogue non-symmetric one. The major issue consists in the summation- and multiplication-type operations which are performed on the kernel matrix. In their naive form, these operations require access the full cross-kernel matrix, and without modification will therefore give rise to the same computational complexity irrespectively of whether a low-rank decomposition of the initial cross-kernel matrix is considered, or not.

We show how this can be circumvented by working exclusively on low-rank factorizations.

Definition 8.7. *Let A be an $(a \times b)$-array. For U an $(a \times r)$-array, and V a $(b \times r)$-array, we say that (U, V) is a low-rank presentation of A, of rank r, if*

$$A[i,j] = \left(U[i,:] \cdot V[j,:]\right)[\Sigma].$$

In matrix notation, this is equivalent to saying that $A = UV^\top$.

Note that in matrix terms, 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:

Lemma 8.8. *Let A be an $(a \times b)$-array with low-rank presentation (U,V). Then:*

(i) *For $m \in \mathbb{N}$, a low-rank presentation of $A[+m,:]$ is $(U[+m,:],V)$. A low-rank presentation of $A[:,+m]$ is $(U,V[+m,:])$.*

(ii) *A low-rank presentation of $A[\boxplus,:]$ is $(U[\boxplus,:],V)$. A low-rank presentation of $A[:,\boxplus]$ is $(U,V[\boxplus,:])$.*

(iii) *A low-rank presentation of $A[\Sigma,:]$ is $(U[\Sigma,:],V)$. A low-rank presentation of $A[:,\Sigma]$ is $(U,V[\Sigma,:])$.*

Proof. The statements follow directly from writing out the decompositions. □

Lemma 8.9. *Let A_1,A_2 be $(a \times b)$-arrays, let U_i be arrays of size $(a \times r_i)$, let V_i be arrays of size $(b \times r_i)$, for $i = 1,2$, for some $r_i \in \mathbb{N}$. Let $A := A_1 + A_2$, write U for the $(a \times (r_1 + r_2))$-array obtained by concatenating U_1,U_2, and V or the $(b \times (r_1 + r_2))$-array obtained by concatenating V_1,V_2 (such that the order of indices matches). Then, the following are equivalent:*

(i) *(U,V) is a low-rank presentation of A, of rank $r_1 + r_2$.*

(ii) *(U_1,V_1) is a low-rank presentation of A_1, of rank r_1, and (U_2,V_2) is a low-rank presentation of A_2, of rank r_2.*

Proof. The statement follows directly from writing out the decompositions of A and $A_1 + A_2$ in terms of U_1,U_2,V_1,V_2 and observing that they are formally equal. □

Notation 8.10. *We introduce notation for index repetition. Let A be a k-fold array of size $(n_1 \times \cdots \times n_k)$.*

(i) *For an integer m, we will write $A[:,\dots,:,m\cdot:,:,\dots,:]$ for the k-fold array of size $(n_1 \times \cdots \times n_{j-1} \times (m \cdot n_j) \times n_{j+1} \times \dots n_k)$ such that $A[:,\dots,:,m\cdot:,:,\dots,:][i_1,\dots,i_{j-1},i_j,i_{j+1},\dots,i_k] = A[i_1,\dots,i_{j-1},r,i_{j+1},\dots,i_k]$, where r is the remainder in integer division of i_j by m. This is intuitively equivalent to concatenating m copies of A along the j-th direction.*

(ii) *For an integer m, we will write $A[:,\dots,:,:\cdot m,:,\dots,:]$ for the k-fold array of size $(n_1 \times \cdots \times n_{j-1} \times (m \cdot n_j) \times n_{j+1} \times \dots n_k)$ such that $A[:,\dots,:,:\cdot m,:,\dots,:][i_1,\dots,i_{j-1},i_j,i_{j+1},\dots,i_k] = A[i_1,\dots,i_{j-1},q,i_{j+1},\dots,i_k]$, where q is the quotient in integer division of i_j by m. This is intuitively equivalent to repeating each slice of A along the j-th direction m times.*

Lemma 8.11. *Let A_1,A_2 be $(a \times b)$-arrays, let U_i be arrays of size $(a \times r_i)$, let V_i be arrays of size $(b \times r_i)$, for $i = 1,2$, for some $r_i \in \mathbb{N}$. Let $A := A_1 \cdot A_2$ (i.e., by our convention, component-wise multiplication), write U for the $(a \times (r_1 \cdot r_2))$-array $U_1[:,:\cdot r_2] \cdot U_2[:,r_1\cdot :]$, and V or the $(b \times (r_1 + r_2))$-array $V_1[:,:\cdot r_2] \cdot V_2[:,r_1\cdot :]$ (multiplication is per convention component-wise). Then, the following are equivalent:*

(i) *(U,V) is a low-rank presentation of A, of rank $r_1 \cdot r_2$.*

(ii) *(U_1,V_1) is a low-rank presentation of A_1, of rank r_1, and (U_2,V_2) is a low-rank presentation of A_2, of rank r_2.*

Proof. The statement follows directly from writing out the decompositions of A and $A_1 \cdot A_2$ in terms of U_1,U_2,V_1,V_2 and observing that they are formally equal. □

Lemmas 8.8, 8.9 and 8.11 cover all operations performed on the kernel matrix in Algorithms 3 and 4, namely, shifting, summation and cumulative summation, component-wise addition and multiplication. Therefore the respective manipulations on low-rank factors may be used to replace all operations within the algorithm. For convenience of the reader, we explicitly present Algorithm 5 which is a low-rank version of Algorithm 3.

Algorithm 5 is obtained from Algorithm 3 as follows: Line 6 is via Lemma 8.8 (i) and (ii). Line 8, is via 8.9 and observing that a low-rank presentation of the all-ones matrix is a pair of all-ones vectors.

Algorithm 5 Evaluation of the sequential kernel k^+_M, with low-rank speed-up
Input: Ordered sequences $\boldsymbol{\sigma},\boldsymbol{\tau}\in\mathcal{X}^+$. A kernel $\mathrm{k}:\mathcal{X}^+\times\mathcal{X}^+\to\mathbb{R}$ to sequentialize. A cut-off degree M.
Output: $\mathrm{k}^+_{(D),M}(\boldsymbol{\sigma},\boldsymbol{\tau})$, as the sequentialization of k

```
1: Let $L\leftarrow\ell(\boldsymbol{\sigma}), L'\leftarrow\ell(\boldsymbol{\tau})$.
2: Compute arrays $U,V$ such that $(U,V)$ is a low-rank presentation of rank $\upsilon$, approximating the kernel
   matrix $K$ with $K[i,j]=k(\nabla\boldsymbol{\sigma}[i],\nabla\boldsymbol{\tau}[j])$
3: (or, alternatively, obtain it as additional input to the algorithm).
4: Initialize an $(M\times L\times *)$-array $B$ and an $(M\times L'\times *)$-array $C$ (where * means that the size may change
   dynamically).
5: Set $B[1|:,:]\leftarrow U$ and $C[1|:,:]\leftarrow V$.
6: for $m=2$ to $M$ do
7:     Compute $P\leftarrow B[m-1|\boxplus+1,:]$ and $Q\leftarrow C[m-1|\boxplus+1,:]$.
8:     Append an $(M\times 1)$-array of ones to $P$, append an $(L'\times 1)$-array of ones to $Q$ .
9:     Set $\rho$ such that $(M\times\rho)$ is the size of $P$, and $(L'\times\rho)$ is the size of $Q$.
10:    Set $B[m|:,:]\leftarrow U[:,:\cdot\rho]\cdot P[:,\upsilon\cdot:]$
11:    Set $C[m|:,:]\leftarrow V[:,\rho\cdot:]\cdot Q[:,:\cdot\upsilon]$
12:    optional: “simplify” the low-rank presentation $(B,C)$, reducing its rank
13: end for
14: Compute $R\leftarrow B[M|\Sigma,:]$
15: Compute $S\leftarrow C[M|\Sigma,:]$
16: Return $1+(R\cdot S)[\Sigma]$
```

Lines 9 and 11 are via 8.11. Lines 12 to 15 are via Lemma 8.8 (iii). Line 13 is evaluation, following the definition of low-rank presentation. Line 10 is optional, aiming at keeping size of the low-rank presentations low (and thus the computational cost); it can be achieved for example via singular value decomposition type techniques, sub-sampling techniques, or random projection type techniques.

The higher order Algorithm 4 can be treated in a similar way, by applying the low-rank representation to the matrices/2D-arrays $A[m|i,j|:,:]$. All assignments and manipulations can be re-phrased in those matrices, therefore the same strategy applies.

The computational cost of one run of Algorithm 5 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 $\ell(\boldsymbol{\sigma})\cdot M\cdot\rho$ space, and C requires $\ell(\boldsymbol{\tau})\cdot M\cdot\rho$ space, then the computational complexity of Algorithm 5 is $O((L+L')\cdot\rho\cdot M)$. Noting that the one can always choose a low-rank representation of $B[m|:,:]$ and $C[m|:,:]$ of rank $\min(L,L')$ or less, the computational complexity of the low-rank algorithm is always bounded by $O(L\cdot L'\cdot M)$, which is the complexity of Algorithm 3.

For the linear kernel, one can infer that the rank will be bounded by $\rho\le n^M$, by using that K admits a low-rank presentation of rank n, then keeping track of matrix sizes: each of the $(M-1)$ repetitions of Lines 9 resp. 11 enlarges the size of B,C by a multiplicative factor of n.

8.4.3. Simultaneous low-rank methods Algorithm 5 yields an efficient low-rank speed-up for computing a single element of the kernel matrix. When employing this speed-up for each entry of the final kernel matrix of size N, the computational cost is $O(N^2\cdot L\cdot\rho\cdot k)$. As stated before, a cost quadratic in N may be prohibitive on large scale data.

This can be addressed by combining both low-rank strategies mentioned before, on sequence-vs-sequence and element-vs-element basis. For this, note that the computation of R depends only on U, and the computation of S depends only on V. If U is chosen to depend only on r, and V only on s, one notes that Algorithm 5 can be split into computation of R an S for each r and s, thus allowing a further reduction of computational cost from $O(N^2\cdot(L+L')\cdot\rho\cdot M)$ to $O(N\cdot(L+L')\cdot\rho\cdot M)$.

Pseudo-code is given in Algorithm 6. It is obtained from Algorithm 5 by starting with a joint low-rank decomposition of element-vs-element kernel matrices, then separating the otherwise redundant computations of the factors R,S.

Algorithm 6 Computation of the sequential kernel matrix k^+_M, with (double) low-rank speed-up
Input: Ordered sequences $\boldsymbol{\sigma}_1,\dots,\boldsymbol{\sigma}_N \in \mathcal{X}^+$. A kernel $\mathrm{k}: \mathcal{X}^+ \times \mathcal{X}^+ \to \mathbb{R}$ to sequentialize. A cut-off degree M.
Output: A matrix U such that (U,U) is a low-rank presentation of the kernel matrix $K \in \mathbb{R}^{N\times N}$ where $K_{ij} = \mathrm{k}^+_M(\boldsymbol{\sigma}_i,\boldsymbol{\sigma}_j)$.

```
 1: Compute arrays $U^{(i)}$ of such that each $(U^{(i)},U^{(j)})$ forms a (joint) low-rank presentation of rank $\upsilon$,
    approximating the element-vs-element kernel matrices $K^{(ij)}$ with $K^{(ij)}[a,b] = \mathrm{k}(\nabla\boldsymbol{\sigma}_i[a],\nabla\boldsymbol{\sigma}_j[b])$
 2: (or, alternatively, obtain it as additional input to the algorithm).
 3: Initialize an $(N \times M \times * \times *)$-array $B$ (where * means that the sizes may change dynamically).
 4: Set $B[i|1|:,:] \leftarrow U^{(i)}$ for all $i \in [N]$.
 5: for $m = 2$ to $M$ do
 6:     Compute $P \leftarrow B[:|m-1|\boxplus{+1},:]$.
 7:     Set $\kappa,\rho$ such that $(N \times \kappa \times \rho)$ is the size of $P$.
 8:     Append an $(N \times \kappa \times 1)$-array of ones to $P$ .
 9:     Set $B[:|m|:,:] \leftarrow B[:|1|:,:\cdot\rho] \cdot P[:,:,\upsilon\cdot:]$
10:     optional: "simplify" the low-rank presentation encoded in $B$, reducing its rank
11: end for
12: Compute $U \leftarrow B[:|M|\Sigma,:]$
13: Return $U$
```

In line 2, the algorithm starts with a joint low-rank presentation of the element-wise kernel matrices - that is, the matrices $U^{(i)}$, when row-concatenated, should have low rank. For example, if k is the Euclidean scalar product, $U^{(i)}$ can be taken as the raw data matrix for s_i, where rows are different time points and columns are features. More generally, jointly low-rank $U^{(i)}$ can be obtained by running a suitable joint diagonalization or singular value decomposition scheme on the element-vs-element kernel matrices $K^{(ij)}$.

Note that such a joint low-rank decomposition may require choice of a higher rank ρ for some kernels k than when only a single entry of the kernel matrix is computed as in Algorithm 6.

8.4.4. Fast sequential kernel methods Following the analogy of sequential kernels to string kernels established in Section 6, fast string kernel methods such as the gappy, substitution, or mismatch kernels presented in [17] 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. The scaling factor λ also may or may not be added in.

It should be noted that not all fast string kernel ideas combine straightforwardly with the low-rank methods introduced above, though they can be adapted. For example, for the gappy kernel, one may consider a joint low-rank decomposition of element-vs-element cross-kernel matrices where suitable entries have been set to zero.

9. Experimental validation

We perform two experiments to validate the practical usefulness of the sequential kernel:

(1) On a real world dataset of hand movement classification (eponymous UCI dataset [31]), we show the sequential kernel outperforms the best previously reported predictive performance [31], as well as non-sequential kernel and aggregate baselines.

(2) On a real world dataset on hand written digit recognition (pendigits), we show that the sequentialization of the Euclidean kernel (= linear use of signature features) achieves only sub-baseline performance, similarly to previously reported results [7]. Using the sequentialized Gaussian kernel improves prediction accuracy to the baseline region.

We would like to stress that our experiments do not constitute a systematic benchmark comparison to prior work, only validation that the sequential kernel is a practically meaningful concept: result (1) validates the first kernelization step in the sense that the order information captured by the sequential kernel can be useful, when compared to alternatives which ignore it. Result (2) validates the second kernelization step in the sense that using a non-linear kernel in sequentialization may outperform the linear kernel.

A 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) which have not been compared to in literature previously.

For the benefit of the scientific community, we have decided to share our more theoretical results early and provide the opportunity to others to work with a toolbox-compatible implementation of the sequential kernels (code link will be provided here shortly), acknowledging that further experimentation is desirable. We will supply benchmark comparisons at a later time point.

9.1. Validation and prediction set-up

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

9.1.2. 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) sequentialization $\mathrm{k}^+_{\leq M}$ of the linear/Euclidean kernel $\mathrm{k}(x,y) = \gamma\langle x,y\rangle$. This sequential kernel has two parameters, a scaling constant γ, and the truncation level M.

(2.b) the (truncated) sequentialization $\mathrm{k}^+_{\leq M}$ of the Gaussian kernel $\mathrm{k}(x,y) = \theta\exp\left(\frac{1}{2}\gamma^2\|x-y\|^2\right)$. This sequential kernel has three parameters: scaling constants γ and θ, and truncation level M.

(1.a) and (1.b) are considered standard kernels, (2.a) and (2.b) are sequential kernels. Note that the non-sequential kernels (1.a) and (1.b) can only be applied to sequential data samples of equal length which is the case for the datasets considered. Note that 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.

We would also like to note a further subtlety: the sequential 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 levels 1 versus 2 and higher, to enable us to make a statement about whether using the order information was beneficial (or not).

There will be no further baseline, naive, or state-of-art predictors in the set-up, comparison will be conducted between the kernel classifiers and performances reported in literature.

We would note that this is a limitation in our set-up which we will rectify in future (more time consuming) instances of a larger benchmarking experiment. The current experiments merely aim to validate whether the sequential kernel is a practically meaningful concept, in particular whether each of the two kernelization steps is practically useful, and whether making use of order information is beneficial.

9.1.3. 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 2 (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 level M	1,2,3
SVC regularizer	0.1, 1, 10, 100, 1000

Table 2: Tuning grid

9.1.4. 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.

9.2. Experiment: Classifying hand movements

We performed classification with the eps-support vector machine (SVC) on the hand movements dataset from UCI [31]. The first database in the dataset 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 9.1, comparing prediction via SVC using one of the following three 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 cut-off level of $M = 1$, corresponding to mean aggregation, and cut-off levels of $M = 2,3$, corresponding to the case where genuine sequence information is used.

The results are reported in Tables 3 to 7. Jackknife standard errors (pooling the five folds) are all 0.04 or smaller. Baseline performance of an uninformed estimator is $1/6 \approx 0.17$.

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, level ≥ 2	0.87	0.86	0.86

Table 3: female1.mat

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, level ≥ 2	0.94	0.97	0.95

Table 4: female2.mat

One can observe that for all five subjects, SVC with sequential kernel of level 2 or higher outperforms SVC using any of the other kernels not using any sequence information.

The sequence kernel appears to outperform the reported methods from the original paper [31] as well (Figures 11 and 12), though this probably may not be entirely clarified due to three issues:

(i) The authors provide no code;

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, level ≥ 2	0.96	0.96	0.96

Table 5: 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, level ≥ 2	0.96	0.96	0.96

Table 6: 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, level ≥ 2	0.93	0.93	0.93

Table 7: male2.mat

(ii) it is not described how the "subject index" in Figures 11 and 12 relates to the subject file names;

(iii) Figures 11 and 12, reporting the results and supposedly pertaining to two different classification methods, are exactly identical, thus likely one of the two is and erroneous copy of the other.

9.3. Experiment: Pendigits

We use performed classification on the pendgits dataset from the UCI repository [8]. 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 level was set to $M = 4$.

The results are reported in Table 8. Jackknife standard errors (pooling the five folds) are all 0.01 or smaller. Baseline performance of an uninformed estimator is $1/10 \approx 0.10$.

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

Table 8: Pendigits

The quality of SVC prediction with the sequentialized linear kernel roughly correspond to those of Diehl [7]. It is outperformed by SVC prediction with the sequentialized Gaussian kernel which is similar to the baseline performance of k-nearest neighbors reported in the documentation of the pendigits dataset.

A. Second kernelization of the signature kernel

We need give meaning to $\mathrm{k}(d\sigma, d\tau)$ when σ takes values in an arbitrary set $\mathcal{X}$ and hence the differentials do not make sense. To motivate the definition below, consider first the case of two paths σ, τ such

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

that $x := \phi(\sigma)$ and $y := \phi(\tau)$ are piecewise linear between time points $\boldsymbol{s} = (\boldsymbol{s}[1],\dots,\boldsymbol{s}[m])$ resp. $\boldsymbol{t} = (\boldsymbol{t}[1],\dots,\boldsymbol{t}[n])$. In this case,

$$\begin{aligned}\int_{(s,t)\in(\boldsymbol{s}[1],\boldsymbol{s}[m])\times(\boldsymbol{t}[1],\boldsymbol{t}[n])} \langle \mathrm{d}x(s), \mathrm{d}y(t)\rangle_{\mathcal{H}} &= \sum_{\substack{i\in[m-1]\\ j\in[n-1]}} \int_{(s,t)\in(\boldsymbol{s}[i],\boldsymbol{s}[i+1])\times(\boldsymbol{t}[j],\boldsymbol{t}[j+1])} \langle \mathrm{d}x(s), \mathrm{d}y(t)\rangle_{\mathcal{H}} \\ &= \sum_{\substack{i\in[m-1]\\ j\in[n-1]}} \langle \nabla x(\boldsymbol{s})[i], \nabla y(\boldsymbol{t})[j]\rangle_{\mathcal{H}} \\ &= \sum_{\substack{i\in[m-1]\\ j\in[n-1]}} \mathrm{k}\begin{bmatrix} \sigma(\boldsymbol{s}[i]) & \sigma(\boldsymbol{s}[i+1]) \\ \tau(\boldsymbol{t}[j]) & \tau(\boldsymbol{t}[j+1]) \end{bmatrix}\end{aligned}$$

where we use the notation

$$\mathrm{k}\begin{bmatrix} a & b \\ c & d \end{bmatrix} := \mathrm{k}(b,d) + \mathrm{k}(a,c) - \mathrm{k}(b,c) - \mathrm{k}(a,d).$$

If $\boldsymbol{s}[1] = \boldsymbol{t}[1] = 0$ and $\boldsymbol{s}[m] = \boldsymbol{t}[n] = 1$, then Proposition 4.5 reads as

$$\mathrm{K}^{\oplus}_{\leq M}(\phi(\sigma),\phi(\tau)) = 1 + \sum_{\substack{i_1\in[m-1]\\ j_1\in[n-1]}} \left(1 + \dots \sum_{\substack{i_M\in[i_{M-1}-1]\\ j_M\in[j_{M-1}-1]}} \mathrm{k}\begin{bmatrix} \sigma(\boldsymbol{s}[i_M]) & \sigma(\boldsymbol{s}[i_M+1]) \\ \tau(\boldsymbol{t}[j_M]) & \tau(\boldsymbol{t}[j_M+1]) \end{bmatrix} \dots \right) \mathrm{k}\begin{bmatrix} \sigma(\boldsymbol{s}[i_1]) & \sigma(\boldsymbol{s}[i_1+1]) \\ \tau(\boldsymbol{t}[j_1]) & \tau(\boldsymbol{t}[j_1+1]) \end{bmatrix}. \tag{A.1}$$

Now define a signed measure on $[0,1]^2$ via the rule

$$\kappa_{\sigma,\tau}([r,s]\times[u,v]) := \mathrm{k}\begin{bmatrix} \sigma(r) & \sigma(s) \\ \tau(u) & \tau(v) \end{bmatrix}$$

and note that (A.1) reads as

$$1 + \int_{[0,1]\times[0,1]} \left(1 + \dots \int_{[0,s_{M-1}]\times[0,t_{M-1}]} \mathrm{d}\kappa_{\sigma,\tau}(s_M,t_M) \dots \right) \mathrm{d}\kappa_{\sigma,\tau}(s_1,t_1).$$

The content of the definition and theorem below is that this formula makes sense for arbitrary paths σ,τ such that $\phi(\sigma),\phi(\tau) \in \mathrm{BV}(\mathcal{H})$.

Definition 3. Let $\mathrm{k} : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ and define the signed-Borel-measure valued map

$$\kappa : \mathrm{Paths}(\mathcal{X}) \times \mathrm{Paths}(\mathcal{X}) \to \mathcal{M}([0,1]\times[0,1]), (\sigma,\tau) \mapsto \kappa_{\sigma,\tau}$$

via $\kappa_{\sigma,\tau}([a,b]\times[c,d]) := \mathrm{k}(\sigma(b),\tau(d)) + \mathrm{k}(\sigma(a),\tau(c)) - \mathrm{k}(\sigma(b),\tau(d)) - \mathrm{k}(\sigma(a),\tau(d))$.

Theorem 4. *Under the Assumptions 4.6, it holds that*

$$\mathrm{k}^{\oplus}_{\leq M}(\sigma,\tau) = 1 + \sum_{m=1}^{M} \int_{(\boldsymbol{s},\boldsymbol{t})\in\Delta_m\times\Delta_m} \mathrm{d}\kappa_{\sigma,\tau}(\boldsymbol{s}[1],\boldsymbol{t}[1]) \cdots \mathrm{d}\kappa_{\sigma,\tau}(\boldsymbol{s}[m],\boldsymbol{t}[m])$$

$$\mathrm{k}^{\oplus}_{\leq M}(\sigma,\tau) = 1 + \int_{(s_1,t_1)\in(0,1)\times(0,1)} \left(1 + \dots \int_{(s_M,t_M)\in(0,s_{M-1})\times(0,t_M-1)} \mathrm{d}\kappa_{\sigma,\tau}(s_1,t_2) \dots \right) \mathrm{d}\kappa_{\sigma,\tau}(s_1,t_2)$$

If $\mathcal{X}$ is an $\mathbb{R}$-vector space and σ,τ are differentiable, then

$$\mathrm{d}\kappa_{\sigma,\tau}(s,t) = \mathrm{k}(\dot{\sigma}(s),\dot{\tau}(t))\,\mathrm{d}s\,\mathrm{d}t.$$

Proof. Let $x := \phi(\sigma)$, $y := \phi(\tau)$ and note that

$$\mathrm{k}^{\oplus}_{\le M}(\sigma,\tau) = \mathrm{K}^{\oplus}_{\le M}(\phi(x),\phi(y)).$$

Fix a sequence $(\boldsymbol{t}_n) \subset \Delta([0,1])$ with mesh $(\boldsymbol{t}_n) \to 0$ as $n \to \infty$ denote with x^n, y^n the paths given by piecewise linear interpolation of points $x(\boldsymbol{t}_n) = (x(\boldsymbol{t}_n[1]),\dots,x(\boldsymbol{t}_n[n]))$, $y(\boldsymbol{t}_n) = (y(\boldsymbol{t}_n[1]),\dots,y(\boldsymbol{t}_n[n]))$. By the above discussion, the statment holds for x^n, y^n if we replace the measure $\kappa_{\sigma,\tau}$ by a measure $\kappa_{\sigma,\tau,n}$ on $[0,1]^2$ as

$$\kappa_{\sigma,\tau,n}([a,b]\times[c,d]) := \langle x^n(b), y^n(d)\rangle_{\mathcal{H}} + \langle x^n(a), y^n(a)\rangle_{\mathcal{H}} - \langle x^n(b), y^n(c)\rangle_{\mathcal{H}} - \langle x^n(a), y^n(d)\rangle_{\mathcal{H}}.$$

As mesh $(\boldsymbol{t}_n) \to 0$, the right hand side converges to $\mathrm{k}\begin{bmatrix}\sigma(a) & \sigma(b)\\ \tau(c) & \tau(d)\end{bmatrix}$. Hence the measure $\kappa_{\sigma,\tau,n}$ converges weakly to the measure $\kappa_{\sigma,\tau}$. On the other hand, $\langle \mathrm{S}(x^n), \mathrm{S}(y^n)\rangle_{\mathrm{T}(\mathcal{H})}$ converges to $\langle \mathrm{S}(x), \mathrm{S}(y)\rangle_{\mathrm{T}(\mathcal{H})}$ which finishes the proof by sending $n \to \infty$ in (A.1). □

B. Integral approximation bounds and proof of Theorem 5

Definition B.1. *Let $[a,b] \subset [0,1]$ and $x \in \mathrm{BV}([a,b],\mathcal{H})$. For $V = [a',b'] \subseteq U$, we write $x[a',b']$ for the element of* $\mathrm{BV}(V,\mathcal{H})$ *which is obtained by restriction of x to V, i.e.,*

$$x[a',b'] := [x : V \to \mathcal{H}] \subseteq \mathrm{BV}(V,\mathcal{H}).$$

The following sum identity becomes very useful.

Proposition B.2. *Let $\boldsymbol{x} \in \mathcal{H}^+$. Then $\mathfrak{S}(\boldsymbol{x}) = \sum_{\boldsymbol{x}' \sqsubset \nabla \boldsymbol{x}} \prod_{i=1}^{\ell(\boldsymbol{x}')} \boldsymbol{x}'[i]$.*

Proof. This follows from an explicit algebraic computation. □

Lemma B.3. *Let $x \in \mathrm{BV}(U)$ for $U = [a,b] \subseteq \mathbb{R}$. Let $m \in \mathbb{N}$, let $V_i := [a_i,b_i] \subseteq U$ for $1 \le i \le m$, and let $V = V_1 \times \cdots \times V_m$. It holds that*

$$\int_V \mathrm{d}x^{\otimes m} = \prod_{i=1}^{m} [x(b_i) - x(a_i)].$$

Proof. Observe that the integral on the left hand side can be split as

$$\int_V \mathrm{d}x^{\otimes m} = \int_{V_1} \cdots \int_{V_m} \mathrm{d}x_1 \otimes \cdots \otimes \mathrm{d}x_m.$$

Separating differential operators, one obtains

$$\int_{V_1} \cdots \int_{V_m} \mathrm{d}x_1 \otimes \cdots \otimes \mathrm{d}x_m = \int_{V_1} \mathrm{d}x_1 \otimes \cdots \otimes \int_{V_m} \mathrm{d}x_m.$$

The claim follows from observing that $\inf_{V_i} \mathrm{d}x_i = x(b_i) - x(a_i)$. □

Lemma B.4. *Let $x \in \mathrm{BV}(U,\mathcal{H})$ for $U = [a,b] \subseteq [0,1]$. It holds that*

$$\left\| \int_{\Delta^m(U)} \mathrm{d}x^{\otimes m} \right\|_{\mathcal{H}} \le \frac{1}{m!} \mathrm{V}(x)^m.$$

Proof. By the (continuous/integral) triangle inequality, it holds that

$$\left\| \int_{\Delta^m(U)} \mathrm{d}x^{\otimes m} \right\|_{\mathcal{H}} \leq \int_{\Delta^m(U)} \|\mathrm{d}x\|_{\mathcal{H}}^m.$$

Further observe that

$$\int_{\Delta^m(U)} \|\mathrm{d}x\|^m = \frac{1}{m!} \int_{U^m} \|\mathrm{d}x\|_{\mathcal{H}}^m.$$

The integral on the right hand side can be split, i.e.,

$$\frac{1}{m!} \int_{U^m} \|\mathrm{d}x\|^m = \frac{1}{m!} \left(\int_U \|\mathrm{d}x\|_{\mathcal{H}} \right)^m.$$

Observing that $\int_U \|\mathrm{d}x\|_{\mathcal{H}} = \mathrm{V}(x)$ yields the claim. □

Lemma B.5. *Fix $m, M \in \mathbb{N}$. Let $x \in \mathrm{BV}([a,b], \mathcal{H})$ and $\boldsymbol{t} = (t_1, \ldots, t_{M+1}) \in \Delta^{M+1}([a,b])$. Write*

$$\mathcal{C} := \int_{\Delta^m(U)} \mathrm{d}x^{\otimes m} \quad \textit{and} \quad \mathcal{D} := \sum_{\substack{\boldsymbol{i} \sqsubset [M] \\ \ell(\boldsymbol{i}) = m}} (\nabla x(\boldsymbol{t}))[i_1] \otimes \cdots \otimes (\nabla x(\boldsymbol{t}))[i_m].$$

Further write, $U := [a,b]$ and $U_i := [t_i, t_{i+1}]$ for $1 \leq i \leq M$, and for $\boldsymbol{i} \sqsubseteq [M]$, write $U_{\boldsymbol{i}} := \bigtimes_{i \in \boldsymbol{i}} [t_i, t_{i+1}]$. Then, the following equalities hold:

(i) $\mathcal{C} - \mathcal{D} = \sum_{\substack{\boldsymbol{i} \sqsubseteq [M] \\ \ell(\boldsymbol{i}) = m \\ \#\boldsymbol{i} \gneq 1}} \int_{\Delta^m(U) \cap U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m}$,

(ii) $\left\| \int_{\Delta^m(U) \cap U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m} \right\|_{\mathcal{H}} \leq \frac{1}{\boldsymbol{i}!} \prod_{i \in \boldsymbol{i}} \mathrm{V}(x[t_i, t_{i+1}])$.

Proof. (i) By splitting the integral, we can write

$$\mathcal{C} = \sum_{\substack{\boldsymbol{i} \sqsubseteq [M] \\ \ell(\boldsymbol{i}) = m}} \int_{\Delta^m(U) \cap U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m}.$$

Note that this is not the same summing over I as in $\mathcal{D}$. In $\mathcal{C}$, indices in the index sequence I may repeat, and for $\mathcal{D}$; they may not as they have to increase strictly monotonously. We will thus write

$$\mathcal{C}_1 := \sum_{\substack{I \sqsubset [M] \\ \ell(\boldsymbol{i}) = m}} \int_{\Delta^m(U) \cap U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m} \quad \text{and}$$

$$\mathcal{C}_2 := \sum_{\substack{I \sqsubseteq [M] \\ \ell(\boldsymbol{i}) = m \\ \boldsymbol{i}! \gneq 1}} \int_{\Delta^m(U) \cap U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m},$$

that is, $\mathcal{C}_1$ collects index sequences without repeated indices, and $\mathcal{C}_2$ collects those with repeat.

Now note that for a non-repeating index sequence I we have $\Delta^m(U) \cap U_{\boldsymbol{i}} = U_{\boldsymbol{i}}$, therefore

$$\mathcal{C}_1 = \sum_{\substack{\boldsymbol{i} \sqsubset [M] \\ \ell(\boldsymbol{i}) = m}} \int_{U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m}$$

Subtraction and collecting terms with the same index yields

$$\mathcal{C}_1 - \mathcal{D} = \sum_{\substack{\boldsymbol{i}\sqsubset[M] \\ \ell(\boldsymbol{i})=m}} \left[\int_{U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m} - (\nabla x(\boldsymbol{t}))[i_1] \otimes \cdots \otimes (\nabla x(\boldsymbol{t}))[i_m] \right].$$

This is zero by Lemma B.3, therefore $\mathcal{C} - \mathcal{D} = \mathcal{C}_1 - \mathcal{C}_2 - \mathcal{D} = \mathcal{C}_2$ which was the claimed statement.

(ii) Fix an index sequence $\boldsymbol{i}$. Let $i_1, \dots, i_k$ the distinct occurring indices in $\boldsymbol{i}$, and $n_1, \dots, n_k$ the total counts of their respective occurrences. Note that therefore

$$\Delta^m(U) \cap U_{\boldsymbol{i}} := \bigtimes_{j=1}^{k} \Delta^{n_j}([\boldsymbol{t}[i_k], \boldsymbol{t}[i_k+1]]).$$

Write $S_j := \Delta^{n_j}([\boldsymbol{t}[i_k], \boldsymbol{t}[i_k+1]])$. Then,

$$\int_{\Delta^m(U)\cap U_{\boldsymbol{i}}} \mathrm{d}x^{\otimes m} = \bigotimes_{j=1}^{k} \int_{S_j} \mathrm{d}x^{\otimes n_j}.$$

Therefore, we obtain as a norm bound

$$\left\| \bigotimes_{j=1}^{k} \int_{S_j} \mathrm{d}x^{\otimes n_j} \right\|_{\mathcal{H}} = \prod_{j=1}^{k} \left\| \int_{S_j} \mathrm{d}x^{\otimes n_j} \right\|_{\mathcal{H}} \le \frac{1}{\boldsymbol{i}!} \prod_{i\in\boldsymbol{i}} \mathrm{V}(x[t_i, t_{i+1}]),$$

where the rightmost inequality follows from applying Lemma B.4 to every multiplicand in the product. □

Theorem 6. *Let $x \in \mathrm{BV}(U, \mathcal{H})$ for $U = [a,b] \subseteq \mathbb{R}$, and let $\boldsymbol{t} \in \Delta^{M+1}(U)$. Write*

$$\mathcal{C} := \mathrm{S}(x) \quad \textit{and} \quad \mathcal{D} := \sum_{\boldsymbol{i}\sqsubset[M]} \prod_{r=1}^{\ell(\boldsymbol{i})} (\nabla x(\boldsymbol{t}))[i_r].$$

Write $\mathcal{C}_m, \mathcal{D}_m$ for the respective homogenous parts of $\mathcal{C}, \mathcal{D}$. Further define

$$G(z) := \exp(z \cdot \mathrm{V}(x)) - \prod_{i=1}^{M} \left(1 + z \cdot \mathrm{V}(x[t_i, t_{i+1}])\right) =: \sum_{m=1}^{\infty} g_m \cdot z^m.$$

That is, the first equality defines G as a function of z, the second defines g_m as the Taylor coefficients of G of its expansion in z around 0.

Then, G is the generating function for an upper bound of approximating $\mathcal{C}$ with $\mathcal{D}$; namely, more precisely:

(i) $\|\mathcal{C}_m - \mathcal{D}_m\|_{\mathcal{H}^{\otimes m}} \le g_m$.

(ii) $\|\mathcal{C} - \mathcal{D}\|_{\mathrm{T}(\mathcal{H})} \le G(1) = \exp(\mathrm{V}(x)) - \prod_{i=1}^{M}\left(1 + \mathrm{V}(x[t_i, t_{i+1}])\right)$.

(iii) If $\boldsymbol{t}$ is chosen such that $\mathrm{V}(x[t_i, t_{i+1}]) = \frac{1}{M}\mathrm{V}(x)$ for all i, then

$$\|\mathcal{C} - \mathcal{D}\|_{\mathrm{T}(\mathcal{H})} \le \frac{\exp(\mathrm{V}(x))}{M} \left(1 + \frac{(\mathrm{V}(x))^M}{(M-2)!}\right).$$

Proof. (i) The bounds given by G are those from Lemma B.5 (i) and (ii) where $\mathcal{C}_m$ here is $\mathcal{C}$ in Lemma B.5 (i), and $\mathcal{D}_m$ here is $\mathcal{D}$ in the lemma. The statement follows from explicitly writing out the coefficient g_m and Lemma B.5 (i) and (ii).

(ii) Applying the triangle inequality and then part (i), one obtains

$$\|\mathcal{C}-\mathcal{D}\|_{\mathrm{T}(\mathcal{H})} \leq \sum_{m=1}^{\infty} \|C_m - \mathcal{D}_m\|_{\mathcal{H}^{\otimes m}} \leq \sum_{m=1}^{\infty} g_m = G(1)$$

and therefore the statement when writing out $G(1)$.

(iii) This follows from observing that for such a choice of $\boldsymbol{t}$, it holds that

$$G(1) = \exp(\mathrm{V}(x)) - \left(1 + \frac{1}{M}\mathrm{V}(x)\right),$$

to which Euler's Theorem 4 may be applied. □

Proposition B.6. *Let $x \in \mathbb{R}, n \in \mathbb{N}$, $x \geq 0$. Let $x_1, \ldots, x_M \in \mathbb{R}$, $x_i \geq 0$ for $i = 1, \ldots, M$ such that $\sum_{i=1}^{M} x_i = x$. Then,*

$$\exp(x) = \prod_{i=1}^{m} (1+x_i) + g(x, x_1, \ldots, x_M), \quad \textit{where } 0 \leq g(x, x_1, \ldots, x_M) \leq x \exp(x) \cdot \max_i x_i.$$

In particular, it holds that

$$\lim_{\max_i x_i \to 0} \prod_{i=1}^{m} (1+x_i) = \exp(x),$$

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. Writing out the product, we obtain

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

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

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

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

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

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

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

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 $\ell(\boldsymbol{i})-1$. On sequences of length m, the map ϕ is at most m-to-one, and surjective onto sequences of length $m-1$. Therefore,

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

where $X=\max_i x_i$. Thus,

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

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

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

□

Note that the bounds from Proposition B.6 are worse than those from Euler's Theorem 4 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.

References

[1] 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.

[2] H. Boedihardjo, X. Geng, T. Lyons, and D. Yang. The Signature of a Rough Path: Uniqueness. *ArXiv e-prints*, June 2014.

[3] Djèlil Chafaï, Terry Lyons, Christoph Ladroue, and Anastasia Papavasiliou. *Computational Rough Paths Project on SourceForge*, 2006 (accessed January 8, 2016).

[4] Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

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

[6] 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.

[7] Joscha Diehl. Rotation invariants of two dimensional curves based on iterated integrals. *CoRR*, abs/1305.6883, 2013.

[8] Peter K Friz and Nicolas B Victoir. *Multidimensional stochastic processes as rough paths: theory and applications*, volume 120. Cambridge University Press, 2010.

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

[10] Benjamin Graham. Sparse arrays of signatures for online character recognition. abs/1308.0371, 2013.

[11] Lajos Gergely Gyurkó, Terry Lyons, Mark Kontkowski, and Jonathan Field. Extracting information from the signature of a financial data stream. *arXiv preprint arXiv:1307.7244*, 2013.

[12] Martin Hairer. A theory of regularity structures. *Inventiones mathematicae*, 198(2):269–504, 2014.

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

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

[15] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97, 2012.

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

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

[18] Daniel Levin, Terry Lyons, Hao Ni, et al. Learning from the past, predicting the statistics for the future, learning an evolving system. *ArXiv e-prints (Sept. 2013)*, 2013.

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

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

[21] Terry 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.

[22] Terry J Lyons. Differential equations driven by rough signals. *Revista Matemática Iberoamericana*, 14(2):215–310, 1998.

[23] Georges Matheron. Principles of geostatistics. *Economic geology*, 58(8):1246–1266, 1963.

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

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

[26] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[27] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[28] Daniel Revuz and Marc Yor. *Continuous martingales and Brownian motion*, volume 293. Springer Science & Business Media, 1999.

[29] 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.

[30] 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.

[31] 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.

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

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

[34] 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.

[35] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. 1996.

[36] Weixin Yang, Lianwen Jin, and Manfei Liu. Character-level chinese writer identification using path signature feature, dropstroke and deep CNN. abs/1505.04922, 2015.