Efficient Path Signature Features

Candidate number 1053510

University of Oxford

CCD Mathematics Dissertation
Word count: 7494 (TeXcount, all words)

Part C Mathematics

Trinity Term 2024

Abstract

When path signatures are used to construct features for machine learning tasks on streamed data, a common approach is to segment the path and compute truncated signatures over each subpath. Choosing the truncation level and the family of subpaths has previously been done in an ad-hoc manner. Finding a feature set which captures enough detail about the paths while not being too high-dimensional involves a trade-off between the truncation level and the number of subpaths.

We study this trade-off from a theoretical and an empirical perspective. For the task of approximating solutions to linear controlled differential equations, we compute error bounds which suggest that in some settings (e.g. when the paths lie in higher dimensional spaces), feature sets which use lower-level signatures over a greater number of subpaths can be more efficient than ones using higher-order signatures. Furthermore, the theory suggests splitting the path not by equal time increments, as is common, but by equal 1-variation. This simple modification substantially increases test accuracy on a real dataset.

Contents

1 Introduction

The signature of a bounded variation path is a collection of all the iterated integrals of the path against itself. While the study of such iterated integrals dates back to the mid-20th century [Che54], in recent years they have found applications in machine learning as a feature transformation for path-like data.

To produce a finite feature set, the signature, which takes values in an infinite-dimensional graded vector space, must be truncated. As the truncation level increases, the size of the truncated signature increases exponentially, so in practice typically only the first few levels are used. To capture further information, previous works (e.g. [Yan+16; Mor+19; Yan+22]) take truncated signature features over multiple, possibly overlapping, subpaths.

Considering a particular feature set of this form, with truncated signatures taken over the subintervals of a partition of the time domain, one can ask: given a desired level of expressivity, which choice of truncation level and partition minimises the dimensionality of the feature set? Increasing either the truncation level or the number of subintervals will increase the feature size, so there is a trade-off between these parameters.

The purpose of this dissertation is to explore this trade-off, and how best to choose the subpaths. This problem has not been previously studied in the literature. While [Mor+20] compares the performance of different ways of partitioning the path, they do not focus on the size of the feature set.

From a theoretical perspective, we compute an error bound for an approximation scheme for solutions to linear CDEs which uses truncated signatures over subintervals, which allows for the comparison of different feature sets. We find that it is optimal to split the path not into equal time intervals, but rather into subpaths of equal 1-variation. In empirical tests, we train simple machine learning models and examine their generalisation performance, using both synthetic and real datasets.

In Sections 2 and 3 we provide a tailored overview of some background material on the theory of rough paths and path signatures. In Section 4 we discuss the use of path signatures in machine learning, and the trade-off we are interested in. In the remainder of the dissertation, we present our attempts to understand the trade-off and the optimal choice of partition: in Section 5 we present our analysis of error bounds for approximations of solutions to CDEs, and in Section 6 we present our empirical tests. Finally, in Section 7 we draw some conclusions.

2 Rough path theory

The goal which rough path theory achieves is to make sense of differential equations driven by paths with low regularity, by enhancing the driving path with additional data. In this section, we establish notation and cover some background material from the theory of rough paths. We follow the approach originally developed by Lyons [Lyo98], and exposited in [LCL07], and we draw heavily from these two sources in this section and in Section 3.

Let V denote a real Banach space with $\dim(V) = d < \infty$.

2.1 Controlled differential equations

Definition 2.1 (p-variation of a path [LCL07, Definition 1.5])**.** For $p \geq 1$ and $X : [0,T] \to V$ a continuous path, define the p-variation of X to be

$$\|X\|_{p,[0,T]} = \left(\sup_{\mathcal{D}} \sum_{i=0}^{r-1} \|x_{t_{i+1}} - x_{t_i}\|^p \right)^{\frac{1}{p}}$$

where the sup is over all partitions $\mathcal{D} = (t_0, \dots, t_r)$ of $[0,T]$. It is not required that mesh $\to 0$ as $n \to \infty$. Let $\mathcal{V}^p([0,T], V))$ denote the space of continuous paths $[0,T] \to V$ of finite p-variation. Paths of finite 1-variation are also called bounded variation paths.

Definition 2.2 (Controlled differential equation)**.** Let V, W be Banach spaces, let $X : [0,T] \to V$ and $Y : [0,T] \to W$ be continuous paths, and let $f : W \to L(V,W)$ be a continuous map from W to the Banach space of bounded linear maps $V \to W$. Then Y is said to satisfy the controlled differential equation (CDE)

$$dY_t = f(Y_t)dX_t, \quad Y_0 = y_0 \tag{1}$$

if

$$Y_t = y_0 + \int_0^t f(Y_s)dX_s \tag{2}$$

for a suitable notion of integral. The map f is called the vector field, and X is called the driving process. If f is linear, that is, $f \in L(W, L(V,W))$, then we call (1) a linear CDE.

Remark 2.3. *For paths of finite* 1*-variation, the integral in (2) can be taken to be the usual Lebesgue-Stieltjes integral. For paths of finite p-variation, with $1 \leq p < 2$, the Young integral can be used – see [LCL07, Section 1.3].*

Theorem 2.4 (Picard-Lindelöf [LCL07, Theorem 1.3])**.** *If X is a bounded variation path and f is Lipschitz continuous, then the CDE (1) has a unique solution.*

The proof of this theorem uses the Picard iteration, just as for the ODE case. Following [LCL07, Section 2.1], we now calculate two Picard iterations in the case that f is linear and bounded, to show how iterated integrals of X arise. The linear map $f : W \to L(V,W)$ can also be considered as a bilinear map $V \times W \to W$, or a linear map $V \to L(W,W)$. With this in mind, in the following we freely switch the order of the arguments of f.

$$\begin{aligned}
Y_t^{(0)} &:= y_0. \\
Y_t^{(1)} &:= y_0 + \int_0^t f(Y_{u_1}^{(0)}) dX_{u_1} \\
&= y_0 + \int_0^t f(y_0) dX_{u_1} \\
&= y_0 + \left(\int_0^t f(dX_{u_1}) \right) (y_0). \\
Y_t^{(2)} &:= y_0 + \int_0^t f(Y_{u_2}^{(1)}) dX_{u_2} \\
&= y_0 + \int_{u_2=0}^{u_2=t} f\left(y_0 + \left(\int_{u_1=0}^{u_1=u_2} f(dX_{u_1}) \right) (y_0) \right) dX_{u_2} \\
&= y_0 + \left(\int_0^t f(dX_u) \right) (y_0) + \left(\int\limits_{0 \le u_1 \le u_2 \le t} f(dX_{u_2}) \circ f(dX_{u_1}) \right) (y_0).
\end{aligned}$$

Now let $V^{\otimes k}$ denote $\underbrace{V \otimes \cdots \otimes V}_{k \text{ times}}$, and define $f^{\otimes k} : V^{\otimes k} \to L(W,W)$ by

$$f^{\otimes k}(x_1 \otimes \cdots \otimes x_k) = f(x_k) \circ \cdots \circ f(x_1) \tag{3}$$

for $x_1, \dots, x_k \in V$, extending by linearity to a map on $V^{\otimes k}$. Note the order reversal, and the fact that $f(x_1), \dots, f(x_k) \in L(W,W)$.

We further define

$$\int\limits_{0<t_1<t_2<\cdots<t_k<T} dX_{t_1} \otimes dX_{t_2} \otimes \ldots \otimes dX_{t_k}$$

to be the iterated integrals of X_t, arranged to form a tensor in $V^{\otimes k}$. To be precise,

$$\int\limits_{0<t_1<\cdots<t_k<T} dX_{t_1} \otimes \ldots \otimes dX_{t_k} = \sum_{i_1,\dots,i_k=1}^{d} \left(\int\limits_{0<t_1<\cdots<t_k<T} dX_{t_1}^{i_1} \cdots dX_{t_k}^{i_k} \right) e_{i_1} \otimes \cdots \otimes e_{i_k}. \tag{4}$$

Then we can write the Picard iterates as functions of the iterated integrals, for example:

$$Y_t^{(2)} = \left(I + f\left(\int\limits_{0<u_1<t} dX_{u_1} \right) + f^{\otimes 2}\left(\int\limits_{0<u_1<u_2<t} dX_{u_1} \otimes dX_{u_2} \right) \right) (y_0). \tag{5}$$

2.2 The tensor algebra

Definition 2.5. The space of formal series of tensors of V is

$$T((V)) = \{(a_0, a_1, a_2, \dots) : a_n \in V^{\otimes n} \text{ for all } n\}.$$

Take $\boldsymbol{x}, \boldsymbol{y} \in T((V))$. An associative product can be defined by

$$(\boldsymbol{x} \otimes \boldsymbol{y})^k = \sum_{i=0}^{k} x^i \otimes y^{k-i} \tag{6}$$

where the superscript k denotes the tensor at index $k \geq 0$.

Definition 2.6. The tensor algebra over V is

$$T(V) = \mathbb{R} \oplus V \oplus V^{\otimes 2} \oplus V^{\otimes 3} \oplus \cdots$$

which can also be equipped with the product given by Equation (6).

While $T((V))$ has elements that are infinite sequences of tensors, elements of the tensor algebra $T(V)$ are finite sequences of tensors.

Definition 2.7. Let $\tilde{T}((V))$ be the affine subspace of $T((V))$ where the first entry is always 1, that is,

$$\tilde{T}((V)) = \{(x_0, x_1, x_2, \dots) \in T((V)) : x_0 = 1\}.$$

When equipped with the product $\otimes$, the space $\tilde{T}((V))$ becomes a group.

Definition 2.8 (Truncated tensor algebra)**.** The truncated tensor algebra at order $n \geq 0$ is

$$T^{(n)}(V) = \mathbb{R} \oplus V \oplus V^{\otimes 2} \oplus \cdots \oplus V^{\otimes n}.$$

The product $\otimes$ is also defined on $T^{(n)}(V)$, by (6) for $0 \leq k \leq n$.

We shall use bold letters to indicate elements of $T((V))$, $T(V)$, or truncated tensor algebras.

Definition 2.9 (Canonical projection map)**.** For $n \geq 0$, define the canonical projection map

$$\pi_n : T((V)) \to T^{(n)}(V), \quad (x_0, x_1, \dots) \mapsto (x_0, x_1, \dots, x_n).$$

We also use π_n to denote the same projection from $T^{(m)}(V) \to T^{(n)}(V)$ when $m \geq n$.

Proposition 2.10. *If* $\dim(V) = d$,

$$\dim\left(T^{(n)}(V)\right) = \frac{d^{n+1} - 1}{d - 1}$$

Proof. Take a basis $\{e_1, \dots, e_d\}$ of V. This naturally gives a basis for $V^{\otimes k}$ for all $k \geq 1$, consisting of the d^k tensors $e_{i_1} \otimes \dots \otimes e_{i_k}$, $i_1, \dots, i_k \in \{1, \dots, d\}$. Trivially, $\{1\}$ is a basis for $\mathbb{R} = V^{\otimes 0}$. Taking the union of these bases for $0 \leq k \leq n$, embedded in $T^{(n)}(V)$, gives a basis for $T^{(n)}(V)$. This basis contains $\sum_{i=0}^{n} d^i = \frac{d^{n+1}-1}{d-1}$ elements. $\square$

Once a basis of V is chosen, one can index coefficients of an element of $T((V))$ using words on an alphabet A of size $\dim(V)$. Let A^* denote the set of all words on A, including the empty word ε. For example, if $A = \{a, b\}$, then

$$A^* = \{\varepsilon, a, b, aa, ab, bb, aaa, \dots\}.$$

If V has dimension 2 with basis e_a, e_b, then a word $w \in A^*$ corresponds to a basis element of $T((V))$, e.g. $aba \leftrightarrow e_a \otimes e_b \otimes e_a$.

Definition 2.11. Given a word w and $\boldsymbol{x} \in T((V))$, or $\boldsymbol{x} \in T^{(n)}(V)$ where $|w| \leq n$, then $\boldsymbol{x}^w$ denotes the coefficient indexed by the word w.

Recall that V is a normed space. Following [LCL07], we assume that the $V^{\otimes k}$ are also equipped with norms which satisfy the following properties.

Definition 2.12 (Admissible norms [LCL07, Definition 1.25])**.** A family of norms on the spaces $V^{\otimes k}$, $k \geq 0$, is called admissible if:

1. The norms are all symmetric: for all $k \geq 1$, for all $x \in V^{\otimes k}$ and all permutations $\sigma \in \mathrm{Sym}(k)$, $\|\sigma x\| = \|x\|$. Here, σx is defined for $x = v_1 \otimes \dots \otimes v_k$ by $\sigma x = v_{\sigma(1)} \otimes \dots \otimes v_{\sigma(k)}$ and extended by linearity to general x.
2. For all $x \in V^{\otimes m}$ and $y \in V^{\otimes n}$,

$$\|x \otimes y\| \leq \|x\| \|y\|. \tag{7}$$

The second property is particularly useful in bounding arguments.

Definition 2.13 (Logarithm in the tensor algebra)**.** Given $\tilde{\boldsymbol{x}} \in \tilde{T}((V))$, write $\tilde{\boldsymbol{x}} = \mathbf{1} + \boldsymbol{x}$, where $x_0 = 0$. Then define the logarithm of $\tilde{\boldsymbol{x}}$ using the usual series

$$\log \tilde{\boldsymbol{x}} = \boldsymbol{x} - \frac{\boldsymbol{x}^{\otimes 2}}{2} + \frac{\boldsymbol{x}^{\otimes 3}}{3} - \frac{\boldsymbol{x}^{\otimes 4}}{4} + \cdots$$

where $\boldsymbol{x}^{\otimes k} = \underbrace{\boldsymbol{x} \otimes \dots \otimes \boldsymbol{x}}_{k \text{ times}}$.

To see that the series converges, note that $(\boldsymbol{x}^i)_j = 0$ for every $i > j \geq 0$, so that the jth level of $\log \tilde{\boldsymbol{x}}$ depends only on the first j terms of the series.

Definition 2.14 (Exponential in the tensor algebra)**.** The exponential function is also defined on $T((V))$ using the usual series,

$$\exp \boldsymbol{x} = \mathbf{1} + \boldsymbol{x} + \frac{\boldsymbol{x}^{\otimes 2}}{2!} + \frac{\boldsymbol{x}^{\otimes 3}}{3!} + \cdots$$

where $\mathbf{1} = (1, 0, 0, \ldots)$.

Convergence can be shown by considering one level of $T((V))$ at a time, using the definition of $\otimes$ given by (6), and the property (7) of admissible norms.

When exp is restricted to $T_0((V)) = \{(x_0, x_1, x_2, \ldots) \in T((V)) : x_0 = 0\}$ and log is defined on $\tilde{T}((V))$, these maps are inverses. Also, the maps exp and log commute with the projections π_n, and hence are well-defined on the truncated spaces [Lyo98, Section 2.1.1.].

2.3 Multiplicative functionals and the extension theorem

Let $\triangle_T$ be the simplex $\{(s,t) \in \mathbb{R}^2 : 0 \le s \le t \le T\}$.

Definition 2.15 (Multiplicative functional [LCL07, Definition 3.1])**.** A multiplicative functional of degree n is a continuous function

$$X : \triangle_T \to \tilde{T}^{(n)}(V)$$

such that, for all $0 \le s \le t \le u \le T$,

$$X_{s,t} \otimes X_{t,u} = X_{s,u}$$

We also call a continuous function $X : \triangle_T \to \tilde{T}((V))$ satisfying this property a multiplicative functional. Multiplicative functionals can be thought of as ordinary paths enhanced with additional information. We now define a notion of regularity for them.

Definition 2.16 (Control function [LCL07, Definition 1.9])**.** A control function is a continuous function $\omega : \triangle_T \to [0, \infty)$ such that $\omega(t,t) = 0$ for all $t \in [0,T]$, and which is superadditive:

$$\omega(s,t) + \omega(t,u) \le \omega(s,u)$$

for all $0 \le s < t < u \le T$.

Definition 2.17 (Multiplicative functional of finite p-variation [LCL07, Definition 3.6])**.** Given $p \ge 1$ (non necessarily an integer) a multiplicative functional $X : \triangle_T \to \tilde{T}^{(n)}(V)$ is said to have finite p-variation controlled by ω if

$$\left\| X^i_{s,t} \right\| \le \frac{\omega(s,t)^{\frac{i}{p}}}{\beta\left(\frac{i}{p}\right)!} \tag{8}$$

for all $i = 1, \ldots, n$ and all $(s,t) \in \triangle_T$, where β is a particular constant depending on p, and $z! = \Gamma(z+1)$ for $z \geq 0$. If such a multiplicative functional exists, we say that X has finite p-variation.

The following is an important result in the theory of rough paths, which states that a multiplicative functional of finite p-variation is determined by its first $\lfloor p \rfloor$ levels.

Theorem 2.18 (The extension theorem [LCL07, Theorem 3.7])**.** *Take $p \geq 1$, an integer $n \geq \lfloor p \rfloor$, and a control function $\omega : \triangle_T \to [0, \infty)$. Let $\boldsymbol{X} : \triangle_T \to \tilde{T}^{(n)}(V)$ be a multiplicative functional of finite p-variation controlled by ω.*

Then for any integer $m \geq n$, there exists a unique multiplicative functional of finite p-variation which extends $\boldsymbol{X}$, i.e. $\boldsymbol{X}^{(m)} : \triangle_T \to \tilde{T}^{(m)}(V)$ satisfies $\pi_n \circ \boldsymbol{X}^{(m)} = \boldsymbol{X}$.

For a full proof, see [Lyo98, Theorem 2.2.1.] or [LCL07, Theorem 3.7]. For our purposes, the most interesting part is the construction of the higher degree multiplicative functional, one level at a time, which we now describe (following [Lyo98] and [LCL07]).

Suppose we have a multiplicative functional of degree m, $\boldsymbol{X}^{(m)}_{s,t} = (1, X^1_{s,t}, \ldots, X^m_{s,t})$, satisfying

$$\left\| X^i_{s,t} \right\| \leq \frac{\omega(s,t)^{\frac{i}{p}}}{\beta(\frac{i}{p})!} \text{ for all } i \leq m. \tag{9}$$

Let $\hat{\boldsymbol{X}}_{s,t} = (1, X^1_{s,t}, \ldots, X^m_{s,t}, 0)$ be the natural embedding of $X_{s,t}$ in $T^{(m+1)}(V)$. Now, given a partition $\mathcal{D} = (t_0, \ldots t_r)$ of $[s,t]$, let

$$\hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t} = \hat{\boldsymbol{X}}_{s,t_1} \otimes \cdots \otimes \hat{\boldsymbol{X}}_{t_{r-1},t_r}.$$

The strategy is to take a sequence of partitions $\mathcal{D}_k$ with mesh $|\mathcal{D}_k| \to 0$, and show that $\hat{\boldsymbol{X}}^{\mathcal{D}_k}_{s,t}$ converges to a multiplicative limit which extends $\boldsymbol{X}^{(m)}$ and satisfies (9) with $i = m+1$.

Given partitions $\mathcal{D}, \tilde{\mathcal{D}}$, let $\hat{\mathcal{D}}$ be their common refinement, and for each interval $[t_i, t_{i+1}]$ in $\mathcal{D}$ let $\hat{\mathcal{D}}_i = \tilde{\mathcal{D}} \cap [t_i, t_{i+1}]$ for $0 \leq i \leq r-1$. Then,

$$\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}}_{s,t} - \hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t} = \hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_0}_{t_0,t_1} \otimes \cdots \otimes \hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_{r-1}}_{t_{r-1},t_r} - \hat{\boldsymbol{X}}_{t_0,t_1} \otimes \cdots \otimes \hat{\boldsymbol{X}}_{t_{r-1},t_r}.$$

This is a pure $(m+1)$-tensor, since for $0 \leq i \leq m$, $(\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}}_{s,t})^i = (\hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t})^i = X^i_{s,t}$ by multiplicativity. Furthermore,

$$(\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}}_{s,t} - \hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t})^{m+1} = \sum_{i=0}^{r-1} \left(\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_j}_{t_i,t_{i+1}} \right)^{m+1}$$

since all the other contributions to level $m+1$ of $\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_0}_{t_0,t_1} \otimes \cdots \otimes \hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_{r-1}}_{t_{r-1},t_r}$ are cancelled by the corresponding contributions to level $m+1$ of $\hat{\boldsymbol{X}}_{t_0,t_1} \otimes \cdots \otimes \hat{\boldsymbol{X}}_{t_{r-1},t_r}$, due to the fact that $\hat{\boldsymbol{X}}^i_{t_j,t_{j+1}} = \left(\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_j}_{t_j,t_{j+1}}\right)^i$ for all $i \leq m$.

Next, we need the following lemma:

Lemma 2.19 (A maximal inequality)**.** *Let X be a multiplicative functional of degree $m \geq \lfloor p \rfloor$ satisfying (9).*

Then for any partition $\mathcal{D}$ of $[s,t]$,

$$\left\| (\hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t})^i \right\| \leq \frac{\omega(s,t)^{\frac{i}{p}}}{\beta(\frac{i}{p})!} \textit{ for all } i \leq m+1. \tag{10}$$

We omit the proof of this lemma. It involves repeatedly removing a carefully chosen point from the partition, and making use of the neoclassical inequality [Lyo98, Theorem 2.2.3.], a generalisation of the binomial theorem.

Now we can bound

$$\begin{aligned}
\left\| \hat{\boldsymbol{X}}^{\hat{\mathcal{D}}}_{s,t} - \hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t} \right\| &\leq \sum_{i=0}^{r-1} \left\| (\hat{\boldsymbol{X}}^{\hat{\mathcal{D}}_j})^{m+1} \right\| \\
&\leq \sum_{i=0}^{r-1} \frac{\omega(t_i,t_{i+1})^{\frac{m+1}{p}}}{\beta\left(\frac{m+1}{p}\right)!} && \text{by Lemma 2.19} \\
&\leq \frac{\omega(0,T)}{\beta\left(\frac{m+1}{p}\right)!} \left(\sup_i \omega(t_i,t_{i+1}) \right)^{\frac{m+1}{p}-1} && \text{using superadditivity of } \omega.
\end{aligned}$$

Since ω is a continuous on the compact set $\triangle_T$, it is uniformly continuous. So, as the mesh of the partition $\mathcal{D}$ goes to zero, $(\sup_i \omega(t_i,t_{i+1}))^{\frac{m+1}{p}-1} \to 0$. By symmetry, the same bound holds with $\mathcal{D}$ replaced by $\tilde{\mathcal{D}}$ and the sup taken over the partition $\tilde{\mathcal{D}}$.

Fix a sequence of partitions $\mathcal{D}_j$ with $\mathrm{mesh}(\mathcal{D}_j) \to 0$. Taking $\mathcal{D} = \mathcal{D}_j$ and $\tilde{\mathcal{D}} = \mathcal{D}_k$, by the triangle inequality we have

$$\left\| \hat{\boldsymbol{X}}^{\mathcal{D}_j}_{s,t} - \hat{\boldsymbol{X}}^{\mathcal{D}_k}_{s,t} \right\| \leq \left\| \hat{\boldsymbol{X}}^{\hat{\mathcal{D}}}_{s,t} - \hat{\boldsymbol{X}}^{\mathcal{D}}_{s,t} \right\| + \left\| \hat{\boldsymbol{X}}^{\hat{\mathcal{D}}}_{s,t} - \hat{\boldsymbol{X}}^{\tilde{\mathcal{D}}}_{s,t} \right\|$$

and applying the bound above to both terms, we see that this can be made arbitrarily small by taking $j,k \geq N$ for sufficiently large N. So, $\left(\hat{\boldsymbol{X}}^{\mathcal{D}_j}_{s,t}\right)$ is a Cauchy sequence in $T^{(m+1)}(V)$. Since V is a Banach space, so is $T^{(m+1)}(V)$, so the sequence converges. The limit can easily be shown to be a multiplicative functional, and since each $\hat{\boldsymbol{X}}^{\mathcal{D}_j}$ satisfies (10), the same inequalities hold for the limit, so it is of finite p-variation. This concludes the extension to a multiplicative functional of degree $m+1$ and finite p-variation.

2.4 Rough paths

While not crucial for what follows, we now make some brief comments on rough paths to provide some more context for the material presented above.

Definition 2.20 (p-rough path)**.** Given $p \geq 1$, a p-rough path is a multiplicative functional of degree $\lfloor p \rfloor$,

$$\boldsymbol{X} : \triangle_T \to T^{(\lfloor p \rfloor)}(V)$$

which is of finite p-variation.

Example 2.21 (Itô and Stratonovich enhanced Brownian motion)**.** Let B_t be a Brownian motion, and let $B_{s,r}$ denote $B_r - B_s$. Let

$$\mathbb{B}^{\text{Itô}}_{s,t} := \int_s^t B_{s,r} \otimes dB_r$$
$$\mathbb{B}^{\text{Strat}}_{s,t} := \int_s^t B_{s,r} \otimes \circ dB_r$$

where the integral for $\mathbb{B}^{\text{Itô}}_{s,t}$ is in the Itô sense, and the integral for $\mathbb{B}^{\text{Strat}}_{s,t}$ is in the Stratonovich sense. It can be shown that for any $p \in (2,3)$, $\boldsymbol{B}^{\text{Itô}} := (1, B_{s,t}, \mathbb{B}^{\text{Itô}}_{s,t})$ and $\boldsymbol{B}^{\text{Strat}} := (1, B_{s,t}, \mathbb{B}^{\text{Strat}}_{s,t})$ are both almost surely p-rough paths [FH20, Proposition 3.4, Proposition 3.5].

A geometric p-rough path [LCL07, Definition 3.13] is a p-rough path that is the limit in a certain sense of a sequence of 1-rough paths, extended to degree $\lfloor p \rfloor$ by the extension theorem. Given a geometric p-rough path $\boldsymbol{Z}$ and a suitably regular function $\alpha : V \to L(V, W)$, it is possible to define an integral

$$\int_s^t \alpha(\boldsymbol{Z}) d\boldsymbol{Z}$$

which can be used to define the notion of a solution to the rough differential equation (RDE). Lyons' Universal Limit Theorem [Lyo98, Theorem 4.1.1] addresses the well-posedness of such differential equations. Details can be found in [LCL07].

3 Path signatures

Continuing with background material, in this section we introduce path signatures and their key properties. In Sections 3.1 through 3.5 we continue to draw heavily from [LCL07].

3.1 Signatures of bounded variation paths

Definition 3.1 (Signature of a path of bounded variation)**.** Let $X \in \mathcal{V}^1([0,T], V)$ be a continuous path of bounded variation. The signature over a subinterval $[s,t] \subseteq$

$[0,T]$, denoted $S(X)_{s,t}$, is an element of $T((V))$ defined by iterated integrals of the path X_t. Using $S^n(X)_{s,t}$ to denote the pure n-tensor at level n, define $S^0(X)_{s,t} = 1$, and for $n \geq 1$,

$$S^n(X)_{s,t} = \int\limits_{s<t_1<t_2<\cdots<t_n<t} dX_{t_1} \otimes dX_{t_2} \otimes \ldots \otimes dX_{t_n}. \tag{11}$$

Recall that the meaning of this integral is given by Equation (4), using the Lebesgue-Stieltjes integral. The theory of such iterated integrals was originally developed by K. T. Chen [Che54; Che57; Che58].

Definition 3.2 (Truncated signature)**.** The level-n truncated signature, denoted $S^{(n)}(X)_{s,t}$ only contains the first $n \geq 0$ levels,

$$S^{(n)}(X)_{s,t} = \pi_n(S(X)_{s,t}).$$

We emphasise that $S^n(X)_{s,t} \in V^{\otimes n}$ is the pure n-tensor at level n, while $S^{(n)}(X)_{s,t} \in T^{(n)}(V)$ contains all tensors up to and including level n of the signature.

Proposition 3.3 (Chen's identity [Che58])**.** *For a path $X : [0,T] \to V$ of bounded variation, the signature is multiplicative, that is,*

$$S(X)_{s,t} \otimes S(X)_{t,u} = S(X)_{s,u}$$

for all $0 \leq s \leq t \leq u \leq T$.

Proof. For $n \geq 1$,

$$S^n(X)_{s,u} = \int_D dX_{r_1} \otimes \ldots \otimes dX_{r_n}$$

where $D = \{s < r_1 < \cdots < r_n < t\}$.

Following [LCL07, Theorem 2.9], we split up the domain of integration and apply Fubini's theorem. For $i = 0, \ldots, n$ let

$$A_i = \{s < r_1 < \cdots < r_i < t\},$$
$$B_i = \{t < r_{i+1} < \cdots < r_n < u\}.$$

Then D can be written as the disjoint union

$$D = \bigcup_{i=0}^{n} A_i \times B_i.$$

Therefore,

$$\begin{aligned}
S^n(X)_{s,u} &= \sum_{i=0}^{n} \int_{A_i \times B_i} dX_{r_1} \otimes \ldots \otimes dX_{r_n} \\
&= \sum_{i=0}^{n} \left(\int_{A_i} dX_{r_1} \otimes \ldots \otimes dX_{r_i} \right) \otimes \left(\int_{B_i} dX_{r_{i+1}} \otimes \ldots \otimes dX_{r_n} \right) \qquad \text{(Fubini)} \\
&= \sum_{i=0}^{n} S^i(X)_{s,t} \otimes S^{n-i}(X)_{t,u} \\
&= \left(S(X)_{s,t} \otimes S(X)_{t,u} \right)^n .
\end{aligned}$$

□

The *shuffle product* is a product on words which is reminiscent of the riffle shuffle from playing cards:

Definition 3.4 (Shuffle product, [FH20, Exercise 2.2]). The shuffle product $v \sqcup\!\sqcup\, w$ between two words is a formal sum of words, defined recursively as follows. If ε is the empty word, then $w \sqcup\!\sqcup\, \varepsilon = \varepsilon \sqcup\!\sqcup\, w = w$. For words v, w and letters a, b, let $va \sqcup\!\sqcup\, wb = (va \sqcup\!\sqcup\, w)b + (v \sqcup\!\sqcup\, wb)a$.

Recall that words can be used to index coordinates of the signature (Definition 2.11). For a formal sum of words, e.g. $v + w$, $S^{v+w}(X)$ denotes $S^v(X) + S^w(X)$.

Proposition 3.5 (Shuffle product property of the signature). *Given $X \in \mathcal{V}^1([0,T], V)$, an alphabet A corresponding to a basis of V, and words $v, w \in A^*$,*

$$S^v(X) S^w(X) = S^{v \sqcup\!\sqcup\, w}(X)$$

Proof ([FH20, Exercise 2.2]). This can be proved by induction on $|v| + |w|$. The inductive step goes through using the product rule for the Lebesgue-Stieltjes integral.

□

Remark 3.6. *We defined the shuffle product on words, but words on* $\dim V$ *letters correspond to basis elements of $T(V^*)$. So, extending by bilinearity, the shuffle product defines an algebra on $T(V^*)$.*

Proposition 3.7 (Factorial decay). *Let $X \in \mathcal{V}^1([0,T], \mathbb{R}^d)$. Then*

$$\|S^n(X)_{0,T}\| \le \frac{\|X\|_{1,[0,T]}^n}{n!}.$$

Proof (based on [LCL07, Proposition 2.2]). Since X is of bounded variation, each component $X^{(i)}$ is the difference of two non-decreasing functions. So, each $X^{(i)}$, and hence X itself, is differentiable almost everywhere. Therefore,

$$\begin{aligned}\left\| \int\limits_{0\le u_1\le\cdots\le u_n\le T} dX_{u_1}\otimes\cdots\otimes dX_{u_n}\right\| &= \left\| \int\limits_{0\le u_1\le\cdots\le u_n\le T} \dot{X}_{u_1}\otimes\cdots\otimes \dot{X}_{u_n} du_1\cdots du_n\right\| \\ &\le \int\limits_{0\le u_1\le\cdots\le u_n\le T} \left\|\dot{X}_{u_1}\right\|\cdots\left\|\dot{X}_{u_n}\right\| du_1\cdots du_n. \end{aligned} \tag{12}$$

The integrand in (12) is invariant under permutations of the variables $u_1,\dots,u_n$. Summing the integrals over all $n!$ permutations gives

$$\begin{aligned} n! \int\limits_{0\le u_1\le\cdots\le u_n\le T} \left\|\dot{X}_{u_1}\right\|\cdots\left\|\dot{X}_{u_n}\right\| du_1\cdots du_n &= \int\limits_{[0,T]^n} \left\|\dot{X}_{u_1}\right\|\cdots\left\|\dot{X}_{u_n}\right\| du_1\cdots du_n \\ &= \left(\int_0^T \left\|\dot{X}_u\right\| du\right)^n \\ &= \|X\|_{1,[0,T]}^n \end{aligned}$$

and the result follows.

□

Next we state that the truncated signature is a continuous map, but first we must define a norm on the vector space $\mathcal{V}^p([0,T],V)$. It is easily checked that the p-variation $\|\cdot\|_{p,[0,T]}$ is a seminorm. It is not a norm, because all constant paths have zero p-variation, but this is easily rectified:

Definition 3.8. Define the *p-variation norm* by

$$\|X\|_{\mathcal{V}^p} = \|X\|_{p,[0,T]} + \sup_{t\in[0,T]} |X_t|$$

This is a norm on the space of continuous paths of finite p-variation.

Proposition 3.9 (Continuity of the truncated signature [LCL07, Corollary 2.11])**.** *With $\mathcal{V}^1([0,T],V)$ equipped with the norm $\|\cdot\|_{\mathcal{V}^1}$, for any integer $n\ge 0$, the truncated signature*

$$\pi_n\circ S : \mathcal{V}^1([0,T],V)\to T^{(n)}(V)$$

is continuous.

Proof. See [Lyo98, Corollary 2.11]. Letting $S_t=\pi_n(S(X))$, the idea is that S_t satisfies a CDE driven by X_t, to which a version of Picard's theorem [LCL07, Theorem 1.28] can be applied to establish continuity.

□

3.2 The signature determines the path

A basic property of the signature, which follows from properties of integration, is that it is invariant under time-reparametrisation. The following definition of tree-like equivalence of paths can be seen as a generalised notion of reparametrisation.

Definition 3.10 (Height function [HL10, Definition 1.2])**.** Given a path $X : [0, T] \to V$ in a normed space, a height function for X is a non-negative continuous function $h : [0, T] \to \mathbb{R}$ such that $h(0) = h(T) = 0$ and

$$\|X_t - X_s\| \leq h(s) + h(t) - 2 \inf_{u\in[s,t]} h(u) \text{ for all } 0 \leq s < t \leq T.$$

Definition 3.11 (Tree-like path [HL10, Definition 1.2])**.** A bounded variation path is called tree-like if it has a height function.

Example 3.12. A path X_t which is injective on $(0, T)$ cannot be tree-like. This is because if X has a height function h, then h must attain a positive maximum $M > 0$ at some $t^* \in (0, T)$. Let $a = \sup\{t \in [0, t^*) : h(t) = \frac{M}{2}\}$ and $b = \inf\{t \in (t^*, T] : h(t) = \frac{M}{2}\}$. Then $0 < a < b < T$, and $h(a) = h(b) = \inf_{s\in[a,b]} h(u)$, therefore $\|X_b - X_a\| = 0$, a contradiction.

Definition 3.13 (Tree-like equivalence [HL10, Definition 1.3])**.** Two bounded variation paths X and Y are said to be tree-like equivalent if the concatenation of X with the reversed path of Y, $X * \overleftarrow{Y}$ is a tree-like path.

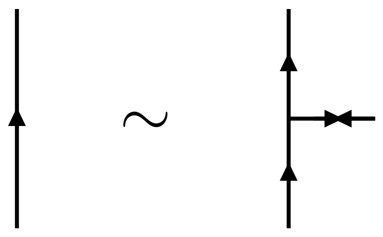

Figure 1: Tree-like equivalent paths

A result of Hambly and Lyons is that the signature determines the path, up to tree-like equivalence.

Theorem 3.14 (Uniqueness of the signature)**.** *Let X and Y be two continuous bounded variation paths. Then X is tree-like equivalent to Y if and only if $S(X) = S(Y)$.*

Proof. Consequence of [HL10, Theorem 4]. □

3.3 The signature is a universal nonlinearity

Recall the classical Stone-Weierstrass theorem:

Theorem 3.15 (Stone-Weierstrass [Sto48])**.** *Let K be a compact Hausdorff space. Let $C(K)$ be the Banach space of continuous real-valued functions on K. Suppose $A \subseteq C(K)$ be a subalgebra which separates points. Then A is dense in $C(K)$.*

This can be used to prove that any continuous real-valued function on paths can be approximated arbitrarily well by a linear function composed with the signature, which motivates the method of performing linear regression on signature features [LLN13].

Theorem 3.16 (The signature is a universal nonlinearity [LLN13, Theorem 3.1])**.** *Let K be a compact subset of the space $\mathcal{V}^1([0,T],\mathbb{R}^d)$ of paths of finite 1-variation, equipped with the norm $\|\cdot\|_{\mathcal{V}^1}$ given by Definition 3.8. Assume that no two paths in K are tree-like equivalent. Let A denote the set of functions $K \to \mathbb{R}$ of the form $X \mapsto \langle \ell, S(X)\rangle$, for $\ell \in T((\mathbb{R}^d)^*)$. Then A is dense in $C(K)$.*

Proof. By assumption K is compact, and as a subset of a metric space it is Hausdorff.

Proposition 3.9 shows that the truncated signature is continuous, and linear functions on $T^{(n)}(\mathbb{R}^d)$ are certainly continuous. Any function in A can be expressed as a linear function of the *truncated* signature, $\langle \ell, S^n(X)\rangle$ for some n, and hence is continuous.

For $c \in \mathbb{R}$, we can set $\ell = (c^*, 0, \ldots)$ where c^* is the constant function which maps every $x \in \mathbb{R}$ to c. So, A contains all constant functions $f(X) \equiv c$. Given two functions $f, g \in A$, we can write $f(X) = \sum_{i=1}^m \alpha_{v_i} S^{v_i}(X)$ and $g(X) = \sum_{j=1}^n \beta_{w_j} S^{w_j}(X)$, where $\alpha_{v_i}, \beta_{w_j} \in \mathbb{R}$ and the words v_i, w_j index signature coefficients (Definition 2.11). By Proposition 3.5,

$$\begin{aligned} f(X)g(X) &= \sum_{i=1}^m \sum_{j=1}^n \alpha_{v_i}\beta_{w_j} S^{v_i}(X) S^{w_j}(X) \\ &= \sum_{i=1}^m \sum_{j=1}^n \alpha_{v_i}\beta_{w_j} S^{v_i \sqcup\!\sqcup w_j}(X). \end{aligned}$$

Therefore $fg \in A$, so A is a subalgebra of $C(K)$.

By Theorem 3.14 and the assumption that no two paths in K are tree-like equivalent, A separates points. By the Stone-Weierstrass theorem, the result follows.

□

Remark 3.17. *The compactness assumption in Theorem 3.16 may seem like a minor technical point, but it is quite restrictive [BPS23, Section 2.2]. It is not clear how we would ensure in practice that this condition is satisfied. Closed balls in the p-variation norm, for example, are not compact. It is possible to remove the compactness assumption and obtain a universal approximation result which applies globally [CST23].*

On the other hand, the assumption that no two paths in K are tree-like equivalent is not so restrictive in practice, as data observed in the real world is unlikely to contain tree-like equivalent paths. It is also easy to prevent distinct paths from being tree-like equivalent by adding time as an additional component.

3.4 Log signatures

The shuffle product property implies that signatures of bounded variation paths contain some algebraic redundancies. For example, $S^i(X)S^j(X) = S^{ij}(X) + S^{ji}(X)$. In particular, the range of the (truncated) signature is a proper subset of the (truncated) tensor algebra. It is common to take the logarithm of the signature, which removes these redundancies:

Definition 3.18 (Log signature)**.** Given a path X of finite 1-variation, the log signature is given by

$$\log(S(X)_{s,t})$$

Definition 3.19 (Truncated log signature)**.** The truncated log signature is simply

$$\log^{(n)} S(X)_{s,t} = \pi_n(\log S(X)_{s,t}).$$

In the next section we shall see that the truncated log signature takes values in a proper linear subspace of the truncated tensor algebra, and in section 3.6 we give the formula for its dimension.

3.5 Free nilpotent Lie groups and Lie algebras

We now seek to understand the range of the truncated signature and log signature maps.

Definition 3.20 (Group-like elements [LCL07, Definition 2.18])**.** Recall from Remark 3.6 that the space $T(V^*)$ can be equipped with the shuffle product to form an algebra.

Define the set $G^{(*)}(V)$ of group-like elements by

$$G^{(*)}(V) = \{\boldsymbol{x} \in \tilde{T}((V)) : \boldsymbol{e}^*(\boldsymbol{x})\boldsymbol{f}^*(\boldsymbol{x}) = (\boldsymbol{e}^* \sqcup\!\sqcup \boldsymbol{f}^*)(\boldsymbol{x}) \text{ for all } \boldsymbol{e}^*, \boldsymbol{f}^* \in T(V^*)\}.$$

For $n \geq 0$, define

$$G^{(n)}(V) = \pi_n(G^{(*)}(V)).$$

Definition 3.21. Let $\mathfrak{g}$ be a Lie algebra, and let U, V be two linear subspaces of $\mathfrak{g}$. Define $[U, V]$ by

$$[U, V] = \text{span}\{[u, v] : u \in U, v \in V\}$$

Definition 3.22 (Lie bracket on $T((V))$)**.** For $x, y \in T((V))$, define

$$[x, y] = x \otimes y - y \otimes x.$$

The vector space $T((V))$ equipped with $[\cdot,\cdot]$ is a Lie algebra.

Definition 3.23 (Lie polynomials)**.** Define the spaces of homogeneous Lie polynomials of degree n, L_n, by $L_0 = 0 \subseteq \mathbb{R}$, $L_1 = V$, and for $n \geq 2$, $L_n = [V, L_{n-1}]$. Define the space of Lie polynomials of degree n as $\mathcal{L}^{(n)}(V) = L_0 \oplus \cdots \oplus L_n$.

Note that $\mathcal{L}^{(n)}(V)$ is a linear subspace of $T^{(n)}(V)$. This is in contrast to $G^{(n)}(V)$, which is a proper subset of $T^{(n)}(V)$ (not all elements of $T^{(n)}(V)$ satisfy the shuffle product property) but not a linear subspace. Instead, $G^{(n)}(V)$ can be thought of as occupying a curved submanifold of $T^{(n)}(V)$, with the exponential map from $\mathcal{L}^{(n)}(V)$ providing a global coordinate chart.

Proposition 3.24. *For $n \geq 0$ and $x \in \tilde{T}^{(n)}(V)$. Then*

$$x \in G^{(n)}(V) \iff \log(x) \in \mathcal{L}^{(n)}(V).$$

Proof. [LCL07, Lemma 2.24]. □

Theorem 3.25 (Range of the truncated signature)**.** *Writing $\mathcal{V}^1 = \mathcal{V}^1([0,T], V)$, the space of continuous bounded variation paths $[0,T] \to V$,*

$$S^{(n)}(\mathcal{V}^1) = G^{(n)}(V).$$

Proof. That $S^{(n)}(\mathcal{V}^1) \subseteq G^{(n)}(V)$ follows from the shuffle product property of the signature (Proposition 3.5).

That the truncated signature is surjective onto $G^{(n)}(V)$ follows from Chow's theorem [FV10, Theorem 7.28], which states that for any $\boldsymbol{x} \in G^{(n)}(V)$ there is a path X such that $S^{(n)}(X) = \boldsymbol{x}$. □

Corollary 3.26 (Range of the truncated log signature)**.**

$$\log S^{(n)}(\mathcal{V}^1) = \mathcal{L}^{(n)}(V).$$

Proof. This follows from Theorem 3.25 and Proposition 3.24. □

3.6 Dimension of the free nilpotent Lie algebra

Let $d = \dim(V)$ be the dimension of the underlying vector space, and let

$$\mathcal{N}_n^d := \dim\left(\mathcal{L}^{(n)}(V)\right).$$

By Corollary 3.26, this is the dimensionality of a truncated log signature, when it is expressed using a basis of $\mathcal{L}^{(n)}(V)$. Since we will be using log signatures to construct features, and we are interested in the dimensionality of our feature sets, we need to know $\mathcal{N}_n^d$.

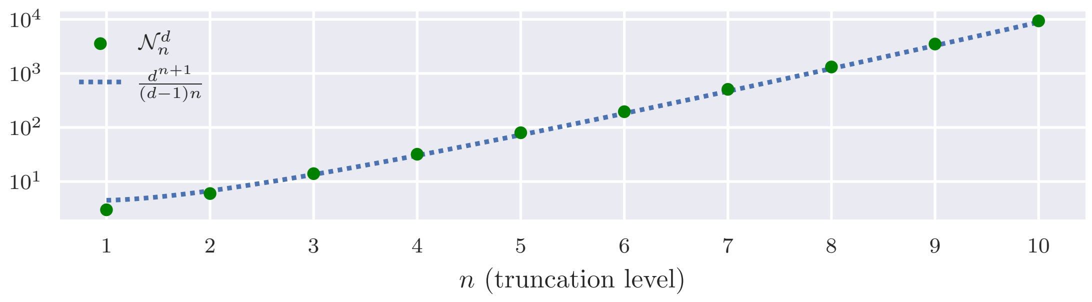

Figure 2: Growth of $\mathcal{N}_n^d$ for $d = 3$.

A common choice of basis for $\mathcal{L}^{(n)}(V)$ is the Lyndon basis, which in fact provides a basis for each L_k that corresponds to the set of Lyndon words on d letters of length k. A Lyndon word is a non-empty word which is strictly lexicographically smaller than any of its proper suffixes, and the mapping from Lyndon words to basis elements is described in [Reu93, Section 4.2] and [Rei17].

Taking for granted that this basis exists, computing $\dim(L_k)$ amounts to counting the number of Lyndon words of length k on d letters. There are d^k words of length $k$on d letters. To count these in a second way, a word of length k can be factorised as k/r repeated copies of a primitive (i.e. not periodic) word of length r. Such a primitive word is associated to a unique Lyndon word of length r by a cyclic rotation. Letting $\ell(r)$ being the number of Lyndon words of length r, such an argument yields

$$d^k = \sum_{r|k} r\ell(r).$$

Applying the Möbius inversion formula from classical number theory gives:

Proposition 3.27 (Witt's formula [Wit37])**.** *For $k \geq 1$, the dimension of L_k is*

$$\ell(k) = \frac{1}{k} \sum_{r|k} \mu\left(\frac{k}{r}\right) d^r$$

where μ is the Möbius function.

Theorem 3.28 (Dimension of the free nilpotent Lie algebra)**.**

$$\mathcal{N}_n^d = \sum_{k=1}^{n} \frac{1}{k} \sum_{r|k} \mu\left(\frac{k}{r}\right) d^r$$

Proof. This follows from Proposition 3.27, since $\mathcal{L}^{(n)}(V) = L_0 \oplus \cdots \oplus L_n$. □

Now, for $n > 1$, $\mathcal{N}_n^d < \dim\left(T^{(n)}(V)\right)$, since the Lyndon words of length at most n are a proper subset of all words of length at most n. However, $\mathcal{N}_n^d$ still grows rapidly as $n \to \infty$. We prove this in the following proposition, building upon the proof [Did] of the asymptotic behaviour of $\sum_{k=1}^{n} \frac{2^k}{k}$.

Proposition 3.29. *When d is fixed, $\mathcal{N}_n^d \sim \frac{d^{n+1}}{(d-1)n}$ as $n \to \infty$.*

Proof. Let $S_n = \sum_{k=1}^{n} \frac{1}{k} d^k$. Recalling that μ takes values in $\{-1, 0, 1\}$, and the second highest divisor of an integer k is at most $k/2$, we have

$$\begin{aligned}
\mathcal{N}_n^d &= S_n + \sum_{k=1}^{n} \sum_{\substack{r|k,\\ r<k}} \mu\left(\frac{k}{r}\right) \frac{1}{k} d^r \\
&= S_n + \sum_{k=1}^{n} \sum_{\substack{r|k,\\ r<k}} O\left(d^{\frac{k}{2}}\right) \\
&= S_n + O\left(n^2 d^{\frac{n}{2}}\right)
\end{aligned}$$

where we have used a crude upper bound of n^2 for the number of terms in the double sum.

Now,

$$\frac{(d-1)n}{d^{n+1}} O\left(n^2 d^{\frac{n}{2}}\right) \to 0 \text{ as } n \to \infty.$$

To deal with the S_n term, we reproduce the argument of [Did], slightly generalised. First,

$$\begin{aligned}
S_n &\geq \sum_{k=1}^{n} \frac{1}{n} d^k \\
&= \frac{1}{n} \left(\frac{d^{n+1} - d}{d-1}\right).
\end{aligned}$$

And for any $u \in (0, 1)$,

$$\begin{aligned}
S_n &= \sum_{k<un} \frac{1}{k} d^k + \sum_{un \leq k < n} \frac{1}{k} d^k \\
&\leq \sum_{k<un} d^k + \sum_{un \leq k < n} \frac{1}{un} d^k \\
&\leq \frac{d^{un} - d}{d-1} + \frac{1}{un} \frac{d^{n+1} - d}{d-1}.
\end{aligned}$$

So,

$$1 - d^{-n} \leq \frac{(d-1)n}{d^{n+1}} S_n \leq n\left(d^{un-(n+1)} - d^{-n}\right) + \frac{1}{u}\left(1 - d^{-n}\right).$$

Taking $n \to \infty$, we obtain

$$1 \leq \liminf_{n\to\infty} \frac{(d-1)n}{d^{n+1}} S_n \leq \limsup_{n\to\infty} \frac{(d-1)n}{d^{n+1}} S_n \leq \frac{1}{u}.$$

Taking $u \uparrow 1$, we obtain $\lim_{n\to\infty} \frac{(d-1)n}{d^{n+1}} S_n = 1$, and the result follows. $\square$

Figure 2 shows that $\mathcal{N}_n^d$ is well approximated by $\frac{d^{n+1}}{(d-1)n}$ even for small n.

4 Path signatures in machine learning

The use of truncated signatures as a feature set for regression analysis with path data was proposed in [LLN13]. Applications for financial data were studied in [Gyu+13; LNO14]. As already noted, this methodology is justified by Theorem 3.16. Signatures have also been used in combination with modern machine learning techniques such as deep learning and gradient boosting, in a range of applications: see, for example, [YJL15; Yan+16; Lia+19; Mor+19; Lia+21; Yan+22]. There are also kernel tricks which allow one to avoid explicitly computing truncated signatures [KO19], and even to implicitly use the untruncated signature [Sal+21].

We consider a framework where truncated signatures, or log signatures, are used to construct a feature set, which can subsequently be used as the inputs for standard machine learning algorithms.

4.1 The trade-off between truncation level and path segmentation

A common technique when creating path signature features is to extract multiple subpaths, compute truncated (log) signatures over each of them, and stack the results to form a feature vector. This is done, for example, in [Yan+16; Mor+19; Yan+22], and [Mor+20] compares a few ways of extracting the subpaths via “windowing” operations. Several works which combine signatures with more sophisticated deep learning architectures also make use of truncated log signatures over subintervals, e.g. [Lia+21; Mor+21; Wal+24].

Taking signatures over m subpaths allows more information to be captured, while only increasing the feature size by a factor of m. In contrast, the feature size grows rapidly as the truncation level is increased (Proposition 2.10; Proposition 3.29). Depending on the dataset, there may be additional reasons to favour features that are localised in time in this way.

The trade-off between truncation level and path segmentation arises when one aims to find a feature set that has minimal dimension while also being sufficiently descriptive. Having low-dimensional features is particularly important in “small data”

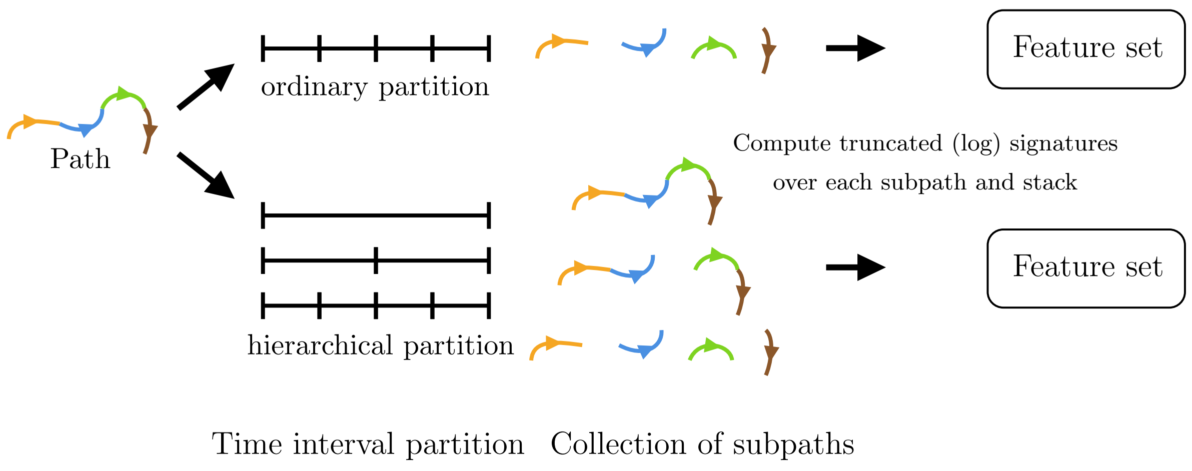

Figure 3: Signature features with ordinary vs. hierarchical partition.

problems, to avoid overfitting. In the following, we consider two ways of measuring how descriptive the features are: the first via error bounds for approximating a certain class of functions, and the second by training models and measuring the generalisation error.

The simplest approach to constructing subpaths is to pick a partition $\mathcal{D} = (t_0, \ldots, t_m)$ of the time domain, and take the collection of level N-truncated signatures over each of the m subintervals $[t_i, t_{i+1}]$. Another approach, illustrated in Figure 3, is to split the time domain in a hierarchical dyadic manner, and compute truncated signatures over every interval (not just those on the lowest level), as in [Yan+16].

Although level-N signatures and log signatures express the same information, given Theorem 3.16 we might expect signatures to express it in a simpler form, in the sense that simpler functions can be used to make predictions with high accuracy. Similarly, by Chen's identity (Proposition 3.3), signatures over the lowest level of the dyadic hierarchy completely determine the signatures over the rest of intervals in the hierarchy, but using the full dyadic hierarchical features can still be beneficial in practice [Mor+20].

Previous works typically split the time domain into equal chunks, but we do not make this restriction. We shall see that the theory suggests splitting by equal length, or equal p-variation instead, and that this can be beneficial on real data.

4.2 Practicalities

We now mention some additional practical considerations that arise when using path signatures to construct features.

- Data is typically given as a series of points, so one must choose how to interpolate to form a path [CK16, Section 2.1].

- There are a number of augmentations that can be applied to paths as pre-processing steps before computing signatures (see [CK16, Section 2.1] and [LM22, Section 2.5]). Augmentations can also be learned [Bon+19]. Several approaches are compared in [Mor+20].
- It is standard practice to normalise features, so that every input variable is on the same scale. In [Mor+20], two dataset-independent approaches to rescaling signature terms are compared. Another option (which we use) is to standardise the features in the typical way, using the dataset statistics.

5 Approximation of solutions to CDEs

In this section, the original contributions of this dissertation begin. We analyse path signature feature sets through error bounds for approximation schemes which use the features to approximate time T solutions to CDEs.

5.1 Linear CDEs

In this subsection, we assume that $\mathbb{R}^d$ is equipped with the ℓ^1 norm and that the tensor product spaces $(\mathbb{R}^d)^{\otimes n}$ are equipped with the projective tensor norm. This can be expressed as

$$\|x\| = \sum_{\substack{i_1,\dots,i_n\\ \in\{1,\dots d\}}} \left|\langle e^*_{i_1}\otimes\cdots\otimes e^*_{i_n}, x\rangle\right| = \sum_{\substack{i_1,\dots,i_n\\ \in\{1,\dots d\}}} \left|x^{i_1,\dots,i_n}\right|. \tag{13}$$

where $e_1,\dots,e_d$ is the standard basis of $\mathbb{R}^d$, and this family of norms is admissible.

Given a continuous bounded variation path, $X \in \mathcal{V}^1([0,T],\mathbb{R}^d)$, consider the linear CDE

$$dY_t = AY_t dX_t, \quad Y_0 = y_0 \tag{14}$$

where Y_t is a path taking values in $\mathbb{R}^e$, and $A \in L(\mathbb{R}^d, L(\mathbb{R}^e,\mathbb{R}^e))$.

Consider the family of functions which map bounded variation paths to the solution of a linear CDE at time $T > 0$:

$$\mathcal{F} = \left\{X \mapsto Y_T : A \in L(\mathbb{R}^d, L(\mathbb{R}^e,\mathbb{R}^e)), y_0 \in \mathbb{R}^e, Y_t \text{ satisfies } (14)\right\}.$$

If the dimension e is allowed to grow arbitrarily large, then functions in $\mathcal{F}$ composed with linear projections form a dense subset of $C(\mathcal{V}^1([0,1],\mathbb{R}^d),\mathbb{R})$. This follows from Theorem 3.16 and the fact that the truncated signature $S^{(n)}$ itself satisfies a linear CDE [LCL07, Lemma 2.10].

We measure the expressivity of a feature map Feat : $\mathcal{V}^1([0,T],\mathbb{R}^d) \to \mathbb{R}^k$ by asking how well functions $f \in \mathcal{F}$ can be approximated by $(g_f \circ \text{Feat})$ for some function

g_f which is allowed to depend on f. In other words, we ask how well the time T solutions of a class of linear CDEs can be approximated using the feature set. Fix constants $L > 0, r > 0, C > 0$. Considering f fixed, we shall assume that the associated A and y_0 are known and can be used to construct g_f. The closeness of $(g_f \circ \mathrm{Feat})$ to f can be measured by

$$\sup_{\substack{X \in \mathcal{V}^1([0,T],\mathbb{R}^d), \\ \|X\|_{1,[0,T]} \le L}} \|f(X) - g_f(\mathrm{Feat}(X))\|$$

and if the feature map is highly expressive, then for all functions $f \in \mathcal{F}$ up to a given regularity, it should be possible to find a g_f such that $(g_f \circ \mathrm{Feat})$ is close to f. So, the expressivity of Feat can be measured by

$$\sup_{\substack{f \in \mathcal{F}, \\ \|A\| \le r, \\ \|y_0\| \le C}} \inf_{g_f} \sup_{\substack{X \in \mathcal{V}^1([0,T],\mathbb{R}^d), \\ \|X\|_{1,[0,T]} \le L}} \|f(X) - g_f(\mathrm{Feat}(X))\| \tag{15}$$

where $\|A\|$ is the operator norm of A as a linear map $\mathbb{R}^d \to L(\mathbb{R}^e, \mathbb{R}^e)$.

We now fix Feat to be the map from a path to the collection of level N log signatures over a partition $\mathcal{D}$ of $[0,T]$ containing m intervals. We choose log signatures in order to minimise the feature size, which is $m\mathcal{N}_N^d$. The function g_f is capable of recovering the truncated signatures if required. We shall consider how to choose $\mathcal{D}$ optimally, and find an upper bound on (15) in terms of m and N.

Lemma 5.1. *For $y \in \mathbb{R}^e$, $x \in (\mathbb{R}^d)^{\otimes n}$,*

$$\left\|A^{\otimes n}(y)(x)\right\| \le \|A\|^n \|y\| \|x\|$$

where $A^{\otimes n}$ is defined as in equation (3).

Proof.

$$\begin{aligned}
\left\|A^{\otimes n}(y)(\boldsymbol{x})\right\| &= \left\| \sum_{\substack{i_1,\dots,i_n \\ \in \{1,\dots d\}}} x^{i_1,\dots,i_n} A^{\otimes n}(y)(e_{i_1} \otimes \cdots \otimes e_{i_d}) \right\| \\
&\le \sum_{\substack{i_1,\dots,i_n \\ \in \{1,\dots d\}}} \left|x^{i_1,\dots,i_n}\right| \left\|A^{\otimes n}(y)(e_{i_1} \otimes \cdots \otimes e_{i_d})\right\| \\
&= \sum_{\substack{i_1,\dots,i_n \\ \in \{1,\dots d\}}} \left|x^{i_1,\dots,i_n}\right| \|A(e_{i_d}) \circ \cdots \circ A(e_{i_1}) y\| \\
&\le \sum_{\substack{i_1,\dots,i_n \\ \in \{1,\dots d\}}} \left|x^{i_1,\dots,i_n}\right| \|A\|^n \|y\| \\
&= \|A\|^n \|y\| \|x\|. && \text{(by (13))}
\end{aligned}$$

□

Proposition 5.2. *The time T solution of the linear CDE (14) is given by*

$$Y_T = \sum_{n=0}^{\infty} A^{\otimes n}(y_0)\left(S^n(X)_{0,T}\right). \tag{16}$$

Proof ([LCL07, Section 2.1]). This follows from the classical Picard iteration, the second iteration of which is given by equation (5). Convergence of the series (16) follows from Lemma 5.1 and Proposition 3.7 (factorial decay).

□

The formula (16) for the time T solution of a linear CDE depends on the full signature. We have seen that the proof of the extension theorem (Theorem 2.18) involves using level-N truncated signatures over a partition to construct the level-$(N+1)$ term of the signature. Motivated by this, we consider approximating higher order terms of the signature from lower order signatures taken over a partition.

Take a partition $0 = t_0 \leq \cdots \leq t_m = T$ of $[0, T]$, and integers $N, k \geq 0$. By repeated applications of Chen's identity (Proposition 3.3),

$$S^{N+k}(X)_{0,T} = \sum_{\substack{n_1,\ldots,n_m \geq 0,\\ n_1+\cdots+n_m = N+k}} S^{n_1}(X)_{t_0,t_1} \otimes \cdots \otimes S^{n_m}(X)_{t_{m-1},t_m}. \tag{17}$$

If k is not too large, then some terms in the sum (17) have $n_i \leq N$ for all i. Given Feat X, these terms are known.

Definition 5.3. Given a partition $\mathcal{D} = (0 = t_0, t_1, \ldots, t_m = T)$, define

$$\tilde{S}^{N,k}_{\mathcal{D}}(X) = \sum_{\substack{0 \leq n_1,\ldots,n_m \leq N,\\ n_1+\cdots+n_m = N+k}} S^{n_1}(X)_{t_0,t_1} \otimes \cdots \otimes S^{n_m}(X)_{t_{m-1},t_m}$$

$\tilde{S}^{N,k}_{\mathcal{D}}(X)$ is the part of the sum (17) which can be computed from the data Feat(X) of truncated signatures over intervals $[t_{j-1}, t_j]$. We consider it as an approximation to $S^{N+k}(X)_{0,T}$.

Observe that

$$\tilde{S}^{N,k}_{\mathcal{D}}(X) = \begin{cases} S^N(X)_{0,T} & \text{if } k = 0, \\ 0 & \text{if } N + k > mN. \end{cases} \tag{18}$$

Proposition 5.4. *Given $X \in \mathcal{V}^1([0,T], \mathbb{R}^d)$ and a partition $\mathcal{D}$ as above,*

$$\left\| S^{N+k}(X)_{0,T} - \tilde{S}^{N,k}_{\mathcal{D}}(X) \right\| \leq \sum_{\substack{n_1+\cdots+n_m = N+k,\\ n_j > N \text{ for some } j}} \frac{\|X\|^{n_1}_{1,[t_0,t_1]} \cdots \|X\|^{n_m}_{1,[t_{m-1},t_m]}}{n_1! \ldots n_m!} \tag{19}$$

Proof.

$$\begin{aligned}\left\|S^{N+k}(X)_{0,T} - \tilde{S}_D^{N,k}\right\| &= \left\| \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} S^{n_1}(X)_{t_0,t_1} \otimes \cdots \otimes S^{n_m}(X)_{t_{m-1},t_m} \right\| \\ &\leq \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \left\|S^{(n_1)}(X)_{t_0,t_1}\right\| \cdots \left\|S^{(n_1)}(X)_{t_{m-1},t_m}\right\| \\ &\leq \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \frac{\|X\|_{1,[t_0,t_1]}^{n_1} \ldots \|X\|_{1,[t_{m-1},t_m]}^{n_m}}{n_1! \ldots n_m!}\end{aligned}$$

using the factorial decay bound (Proposition 3.7). □

Since X is continuous,

$$\|X\|_{1,[0,T]} = \sum_{i=0}^{m-1} \|X\|_{1,[t_i,t_{i+1}]}$$

so we can minimise the bound (19) by solving the optimisation problem

$$\begin{aligned} \min \quad & \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \frac{x_1^{n_1} \ldots x_m^{n_m}}{n_1! \ldots n_m!} \\ \text{s.t.} \quad & x_i > 0, \quad i = 1, \ldots, m \\ & x_1 + \cdots + x_m = L \end{aligned} \tag{20}$$

where $L = \|X\|_{1,[0,T]}$, and the inequalities $x_i > 0$ are taken to be strict to ensure that the partition genuinely has m nonempty subintervals.

Remark 5.5. *When $N+k > mN$, the constraint in the sum that $n_j > N$ for some j is redundant, and*

$$\sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \textit{ for some } j}} \frac{x_1^{n_1} \ldots x_m^{n_m}}{n_1! \ldots n_m!} = (x_1 + \cdots + x_m)^{N+k} = L^{N+k}$$

which is independent of the choice of the x_i. Next, we prove that when $N+k \leq mN$, the optimisation problem (20) is solved by taking $x_1 = \cdots = x_m = \frac{L}{m}$. It is tempting to simply appeal to the symmetry of the problem, but since this objective function is in general non-convex, this does not immediately give the conclusion.

Proposition 5.6. *Suppose $L > 0$ and $N \geq 1$, $m \geq 1$, $k \geq 1$ are given such that $N + k \leq mN$. Then the solution to the minimisation problem (20) is given by $x_1 = \cdots = x_m = \frac{L}{m}$.*

Proof. If $x_1, \ldots, x_m > 0$ satisfy $x_1 + \cdots + x_m = L$ and are not all equal, then without loss of generality we may assume that $0 < x_1 < x_2$. Consider setting $\hat{x}_1 = x + s$, $\hat{x}_2 = x - s$, and $\hat{x}_j = x_j$ for all $j \geq 3$, where $s \in [0, \hat{x}_2 - \hat{x}_1]$. These $\hat{x}_i$ still satisfy the constraints of (20). Let $F(s)$ be the value of the objective function evaluated at $\hat{x}_1, \ldots, \hat{x}_m$:

$$F(s) = \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \frac{(x_1+s)^{n_1}(x_2-s)^{n_2} x_3^{n_3} \ldots x_m^{n_m}}{n_1! \ldots n_m!}.$$

Then

$$\begin{aligned} F'(s) = &\sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \frac{n_1(x_1+s)^{n_1-1}(x_2-s)^{n_2} x_3^{n_3} \ldots x_m^{n_m}}{n_1! \ldots n_m!} \\ &- \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \frac{n_2(x_1+s)^{n_1}(x_2-s)^{n_2-1} x_3^{n_3} \ldots x_m^{n_m}}{n_1! \ldots n_m!}. \end{aligned}$$

Evaluating at zero:

$$\begin{aligned} F'(0) = &\sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j,\\ n_1 \geq 1}} \frac{n_1 {x_1}^{n_1-1} {x_2}^{n_2} x_3^{n_3} \ldots x_m^{n_m}}{n_1! \ldots n_m!} \\ &- \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j,\\ n_2 \geq 1}} \frac{n_2 {x_1}^{n_1} {x_2}^{n_2-1} x_3^{n_3} \ldots x_m^{n_m}}{n_1! \ldots n_m!} \\ = &\sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j,\\ \tilde{n}_1 := n_1-1 \geq 0}} \frac{{x_1}^{\tilde{n}_1} {x_2}^{n_2} x_3^{n_3} \ldots x_m^{n_m}}{\tilde{n}_1! n_2! \ldots n_m!} \\ &- \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j,\\ \tilde{n}_2 := n_2-1 \geq 0}} \frac{{x_1}^{n_1} {x_2}^{\tilde{n}_2} x_3^{n_3} \ldots x_m^{n_m}}{n_1! \tilde{n}_2! n_3! \ldots n_m!} \\ = &\sum_{A_1} \frac{x_1^{n_1} \ldots x_m^{n_m}}{n_1! \ldots n_m!} - \sum_{A_2} \frac{x_1^{n_1} \ldots x_m^{n_m}}{n_1! \ldots n_m!}, \end{aligned} \tag{21}$$

where in the final inequality (21) we have reset the indices $n_1 := \tilde{n}_1$ and $n_2 := \tilde{n}_2$ respectively, with

$$A_i := \{n_1, \ldots, n_m \geq 0 : \sum_i n_i = N+k-1 \text{ and } (n_j > N \text{ for some } j \neq i, \text{ or } n_i > N-1)\}.$$

Now, we can decompose A_i as a disjoint union of sets, $A_i = B_i \cup C_i$ where

$$C_i = \{n_1 + \ldots + n_m = N + k - 1, n_j > N \text{ for some } j\},$$
$$D_i = \{n_1 + \ldots + n_m = N + k - 1, n_i = N \text{ and } n_j \leq N \text{ for all } j \neq i\}.$$

Since C_i does not actually depend on i, the contributions from C_1 and C_2 in (21) cancel, and we are left with

$$F'(0) = \sum_{D_1} \frac{x_1^{n_1} \dots x_m^{n_m}}{n_1! \dots n_m!} - \sum_{D_2} \frac{x_1^{n_1} \dots x_m^{n_m}}{n_1! \dots n_m!}.$$

We claim that $F'(0) < 0$. We have

$$\begin{aligned} F'(0) &= \frac{{x_1}^N}{N!} \sum_{\substack{n_2+\dots+n_m=k-1,\\ n_j \le N \text{ for all } j}} \frac{{x_2}^{n_2} \dots {x_m}^{n_m}}{n_2! \dots n_m!} - \frac{{x_2}^N}{N!} \sum_{\substack{n_1+n_3+\dots+n_m=k-1,\\ n_j \le N \text{ for all } j}} \frac{{x_1}^{n_1} {x_3}^{n_3} \dots {x_m}^{n_m}}{n_1! n_3! \dots n_m!} \\ &= \sum_{n=0}^{N} \frac{{x_1}^N {x_2}^n}{N! n!} \sum_{T_n} \frac{{x_3}^{n_3} \dots {x_m}^{n_m}}{n_3! \dots n_m!} - \sum_{n=0}^{N} \frac{{x_2}^N {x_1}^n}{N! n!} \sum_{T_n} \frac{{x_3}^{n_3} \dots {x_m}^{n_m}}{n_3! \dots n_m!}, \end{aligned} \tag{22}$$

where

$$T_n = \{(n_3, \dots, n_m) : n_3 + \dots + n_m = k - 1 - n, 0 \le n_j \le N \text{ for all } j \ge 3\}$$

and we note that T_n may be empty for some choices of n. Since $x_1 < x_2$, for each $0 \le n \le N$ we have ${x_1}^N {x_2}^n \le {x_2}^N {x_1}^n$. The inequality is strict when $n < N$, and in particular when $n = 0$. The inner sums of (22) are the same for each n, and are strictly positive unless T_n is empty, in which case they are zero. Now, $N + k \le mN$ implies that $(m - 1)N \ge k$, which implies that T_0 is nonempty. Overall this shows that for each $n = 0, \dots, N$,

$$\frac{{x_1}^N {x_2}^n}{N! n!} \sum_{T_n} \frac{{x_3}^{n_3} \dots {x_m}^{n_m}}{n_3! \dots n_m!} \le \frac{{x_2}^N {x_1}^n}{N! n!} \sum_{T_n} \frac{{x_3}^{n_3} \dots {x_m}^{n_m}}{n_3! \dots n_m!},$$

with strict inequality when $n = 0$. Therefore, $F'(0) < 0$, so unless $x_1 = \ldots = x_m = \frac{L}{m}$ in (20), a small perturbation can decrease the objective while still satisfying the constraints. The claim is proved.

□

Proposition 5.6 shows that when $N + k \le mN$, to minimise (19) it is optimal to choose the partition $\mathcal{D}$ such that each part has equal 1-variation. Proposition 5.4 then gives:

$$\begin{aligned} \left\| S^{N+k}(X)_{0,T} - \tilde{S}^{N,k}_{\mathcal{D}}(X) \right\| &\le \sum_{\substack{n_1+\dots+n_m=N+k,\\ n_j > N \text{ for some } j}} \frac{\left(\frac{L}{m}\right)^{n_1} \dots \left(\frac{L}{m}\right)^{n_m}}{n_1! \dots n_m!} \\ &= \left(\frac{L}{m}\right)^{N+k} \frac{C(N,k,m)}{(N+k)!} \end{aligned} \tag{23}$$

where

$$C(N,k,m) := \sum_{\substack{n_1+\cdots+n_m=N+k,\\ n_j>N \text{ for some } j}} \binom{N+k}{n_1,\ldots,n_m}.$$

When $N+k > mN$, $\tilde{S}^{N,k}_{\mathcal{D}}(X) = 0$ and $C(N,k,m) = m^{N+k}$, so we recover the a priori bound of Proposition 3.7.

Definition 5.7. By plugging the approximation $\tilde{S}^{N,k}_{\mathcal{D}}(X)$ into (16), and using (18), we obtain an approximation to the time T solution:

$$\tilde{Y}_T := \sum_{n=0}^{N} A^{\otimes n}(y_0)\left(S^n(X)_{0,1}\right) + \sum_{n=N+1}^{mN} A^{\otimes n}(y_0)\left(\tilde{S}^{N,n-N}_{\mathcal{D}}(X)\right).$$

Note that this is a function of $\operatorname{Feat}(X)$, the level N signatures over subintervals of $\mathcal{D}$.

Proposition 5.8. *Let $r>0$ and $L>0$ be constants. For any linear vector field A with $\|A\| \le r$, let $\tilde{Y}_T$ be defined as above, with the partition splitting the path into parts of equal 1-variation. Then for any driving path $X : [0,T] \to \mathbb{R}^d$ with $\|X\|_{1,[0,T]} \le L$,*

$$\left\|Y_T - \tilde{Y}_T\right\| \le \sum_{n=N+1}^{mN} \left(\frac{rL}{m}\right)^n \frac{C(N,n-N,m)}{n!}\|y_0\| + \sum_{n=mN+1}^{\infty} (rL)^n \frac{1}{n!}\|y_0\| \qquad (24)$$

Proof.

$$Y_T - \tilde{Y}_T = \sum_{n=N+1}^{mN} A^{\otimes n} y_0 \left(S^n(X)_{0,T} - \tilde{S}^{N,n-N}_{\mathcal{D}}(X)\right) + \sum_{n=mN+1}^{\infty} A^{\otimes n} y_0 S^n(X)_{0,T}$$

Then take norms and apply Lemma 5.1, the bound (23), and the a priori factorial decay bound of Proposition 3.7. □

Taking g_f to be the function which maps $\operatorname{Feat}(X)$ to the approximation $\tilde{Y}_T$, (24) amounts to an upper bound on (15), as desired. If N, m, and rL are known, then the value of (24) can be calculated exactly. The infinite series term is the tail of an exponential series, and the combinatorial quantity $C(N,k,m)$ can be efficiently calculated using recursion in m (see the Appendix for code). These error bounds are plotted in Figure 4. One can read off from these plots the truncation level of the most efficient feature set that achieves a given value for the error bound. When $d=2$, higher values of N tend to be optimal. When d is increased to 3 or 4, higher truncation levels are more costly in terms of feature size (recall Proposition 3.29) and we see that lower values of N, e.g. $2,3,4$, are optimal. Increasing rL also favours features sets with lower N and higher m, due to the appearance of $\left(\frac{rL}{m}\right)^n$ in (24).

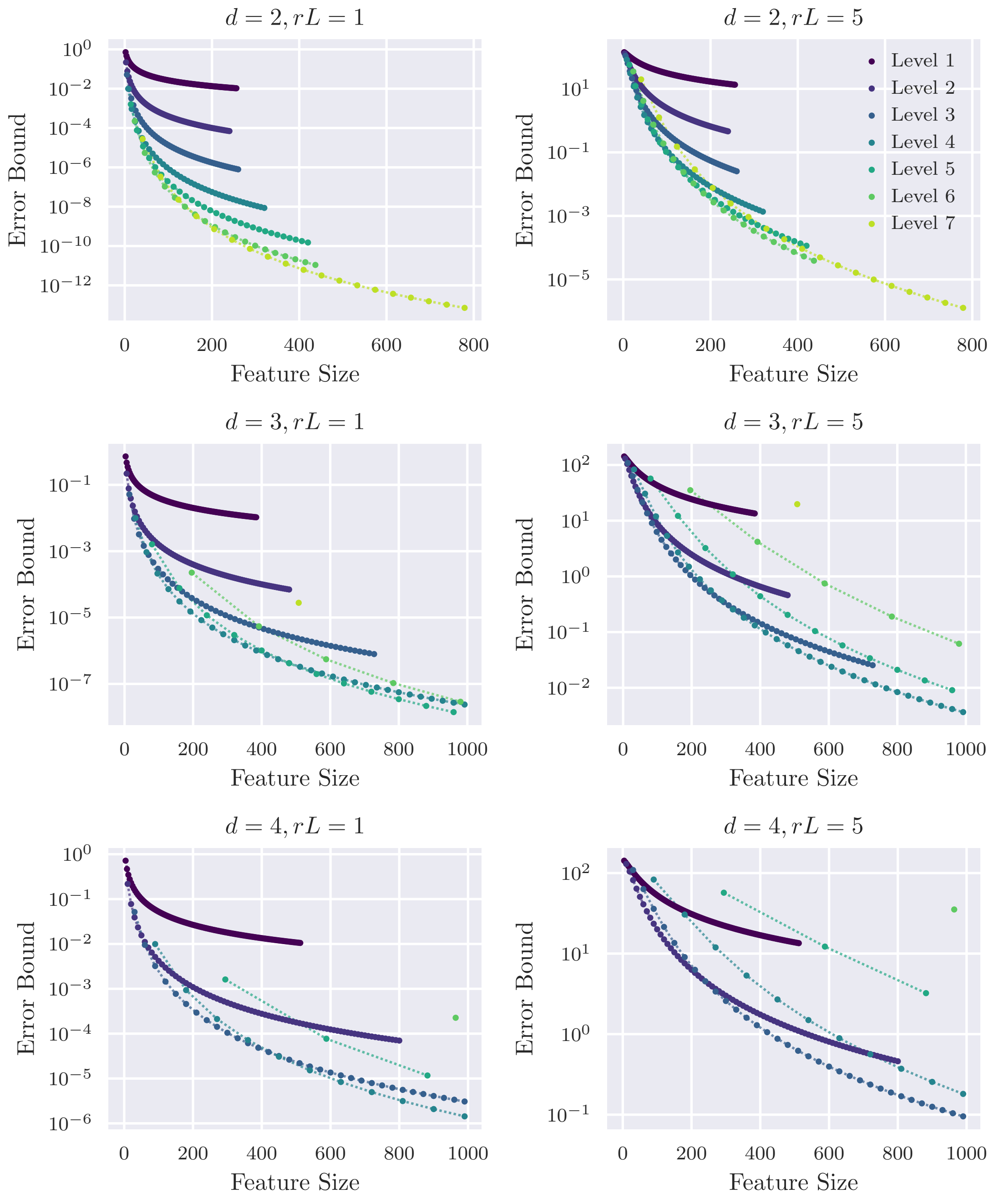

Figure 4: Plots of the upper bound (24) from Proposition 5.8 against the size $m\mathcal{N}_N^d$ of the log signature feature set, where dimension $d \in \{2, 3, 4\}$ and $rL \in \{1, 5\}$. For each truncation level N, the number of subintervals m increases in steps of 1 up to a maximum determined by computational constraints. We assume $\|y_0\| = 1$.

Remark 5.9. *There are specific functions in $\mathcal{F}$ for which a given error tolerance is achieved most efficiently using higher order signature terms. For example, the truncated signature itself is a solution to a linear CDE. Crucially, the error bounds plotted in Figure 4 apply for all linear vector fields A with $\|A\| \leq r$.*

5.2 Nonlinear CDEs

The approximation given in Definition 5.7 can also be derived using truncations of the series (16) in a "multi-step" approximation method with respect to the partition $\mathcal{D} = (t_0, \ldots, t_m)$. Let $\tilde{Y}^{\text{multi}}_{t_0} = y_0$ and for $i = 0, \ldots, m-1$ let

$$\tilde{Y}^{\text{multi}}_{t_{i+1}} = \sum_{n=0}^{N} A^{\otimes n}(\tilde{Y}^{\text{multi}}_{t_i})(S^n(X)_{t_i,t_{i+1}}).$$

Then

$$\tilde{Y}^{\text{multi}}_{t_m} = \tilde{Y}_{t_m}.$$

Furthermore, this is a special case of the "step-N Euler scheme" defined in [FV10, Definition 10.29], which is also defined when the vector field is nonlinear, and converges as long as the vector field is sufficiently regular.

As in the previous subsection, the scheme only uses data about the driving path in the form of level-N signatures over subintervals of a partition, and produces an estimate of the time T solution. We mention this because the form of the error bound for the Euler scheme again suggests choosing the partition using an "equal p-variation" split. So far we have only considered signatures of bounded variation paths, but signatures can also be defined for paths of finite p-variation for $p \in [1, 2)$ (e.g. using the extension theorem). We state the following proposition for $p \in [1, 2)$, although it is a special case of a result which holds for all $p \geq 1$.

Proposition 5.10. *Suppose X is a path of finite p-variation, where $p \in [1, 2)$, and $N \geq 1$. For a sufficiently regular vector field (the required level of regularity depends on N), the step-N Euler scheme over the partition $\mathcal{D} = (t_0, \ldots, t_m)$ has the error bound*

$$\left\| Y_T - Y_T^{\mathit{Euler};\mathcal{D}} \right\| \leq C \sum_{i=1}^{m} \|X\|^{N+1}_{p,[t_{i-1},t_i]} \tag{25}$$

where C is a constant depending on N, p, the vector field, and $\|X\|_{p,[0,T]}$.

Proof. This follows from [FV10, Theorem 10.30]. The statement there involves a notion of p-variation for paths taking values in $G^{(\lfloor p \rfloor)}(\mathbb{R}^d)$ based on the Carnot-Carathéodory norm, but when $p \in [1, 2)$, the Carnot-Carathéodory norm of $X_{t_i,t_{i+1}}$ is simply $\|X_{t_i,t_{i+1}}\|_{\mathbb{R}^d}$, so that the notion of p-variation agrees with Definition 2.1. □

If $p = 1$, then since $\sum_{i=1}^{m} \|X\|_{1,[t_{i-1},t_i]} = \|X\|_{1,[0,T]}$ is fixed and $N+1 > 1$, the bound (25) is minimised by choosing the partition such that the $\|X\|_{1,[t_i,t_{i+1}]}$ are all equal. In contrast to (20) this is immediate by convexity.

Remark 5.11. *The log-ODE method is another scheme which uses the same data and has an error bound similar to (25) [Mor+21, Theorem B.7].*

6 Empirical tests of generalisation performance

6.1 Methodology

We now train simple machine learning models using path signature features, taken over subpaths. Our goal is to compare choices of N, the truncation level, and m, the number of subintervals[1] to see what is the most efficient way to achieve a given test accuracy. We consider ordinary partitions and (in Section 6.2 only) hierarchical partitions (recall Figure 3). We consider splitting into equal time segments, or, motivated by the previous section, into equal 1-variation segments, or equal p-variation for some $p > 1$. This approach to splitting maintains the invariance of the features under time-reparametrisation. It is not computationally expensive, since it only requires computing the running p-variation (linear time in the length of the path) and finding the points at which to split using $m - 1$ binary searches.

Given level-N log signatures expressed, for example, in the Lyndon basis, the coefficients of the level N signature can be recovered using polynomial functions of the log signature coefficients. If these are given over subintervals in a partition, tensoring the truncated signatures together is also a polynomial function. So, by Theorem 3.16, this gives a universal approximation result for polynomial functions of truncated log signature features over subintervals (when the truncation level is allowed to grow arbitrarily large).

Motivated by this, for regression we use kernel ridge regression with a polynomial kernel, and for classification we use support vector machines with a polynomial kernel. The advantage of using kernel methods is that they do not require explicitly computing the polynomial features.

We perform experiments on two synthetic datasets and one real dataset. In each case, 25% of the data is withheld to create a test set. The feature sets are constructed using RoughPy[2] [ML24] to compute (log) signatures. For normalisation, we apply min-max scaling to the paths before taking signatures, and standardise again after taking signatures, such that every feature has zero mean and unit variance. We use the scikit-learn [Ped+11] implementations of kernel ridge regression and support vector machines, and run a 5-fold cross-validation procedure over the training set to optimise the degree of the polynomial kernel and the regularisation parameter. The

[1] [Lia+19, Figure 8] does this for the logsig-RNN architecture.

[2] We thank Sam Morley for a helpful conversation in which he explained the interface.

γ parameter is set using the scikit-learn default. Using the optimal hyperparameters, we retrain the model on the full training set and evaluate it on the test set.

6.2 Synthetic data: linear CDE solution

In this experiment, we generate a synthetic dataset by sampling 2000 paths of a simple symmetric random walk in $\mathbb{R}^2$, with 128 time steps. The sizes of the jumps are scaled such that the total length of each path is 5. Note that such paths are already parameterised by (a constant multiple of) their 1-variation.

The target associated with each path is the solution $Y_1 \in \mathbb{R}^2$ of the linear CDE

$$dY_t = A(Y_t)dX_t, \quad Y_0 = (0.5, 0.5)$$

where the vector field A is chosen by taking a $2 \times 2 \times 2$ array with Gaussian random entries and rescaling such that the $\|A\| = 1$ (in the same sense as in Section 5.1, recalling that the spaces $\mathbb{R}^2$ are equipped with the $\|\cdot\|_1$ norm). The solution is computed numerically using Diffrax [Kid21] with the Tsit5 solver. We observe that if the values of $\|A\|$ or the path length are set too small, then Y_1 can be very well approximated simply by the total increment of the driving path X. We choose values such that this is not the case. With this setup, we can compute the error bound (24), given N and m.

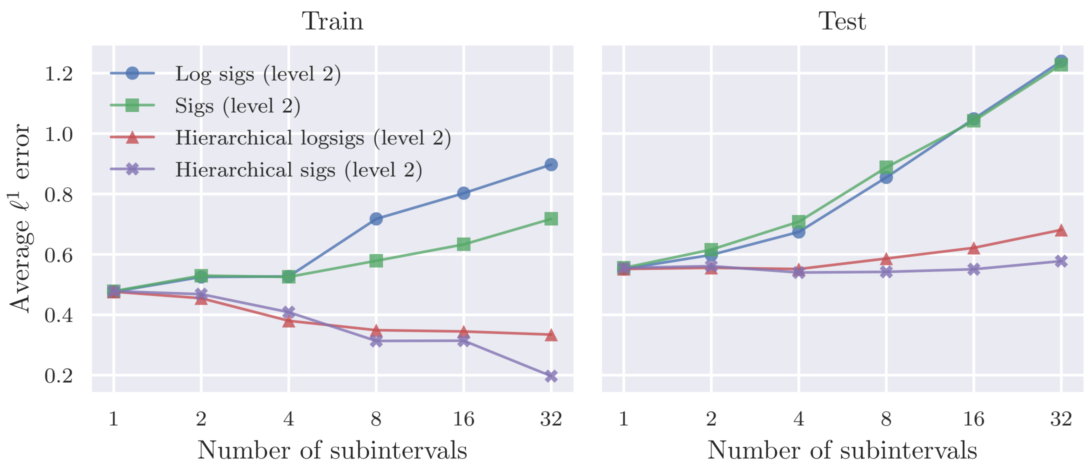

Figure 5: Comparison of the average ℓ^1 distance from predicted to actual solution (lower is better) on train and test sets for various feature set constructions.

We train kernel ridge regression models using the methodology described above, for feature sets using either signature or log signatures, and either an ordinary or hierarchical partition. In each case, we split the time domain into equal subintervals. When the ordinary partition is used, the test error increases with the number of subintervals, despite the feature set containing more information. Figure 5 shows this for $N = 2$. This is not purely due to overfitting, since the train error also increases. A possible explanation for this effect is that using more subintervals requires a more complicated function of the features to be learned – observe that truncating (16) gives an approximation for Y_1 as a linear function of the signature over the entire path, while the same approximation expressed in terms of signatures over subpaths requires tensoring them together, resulting in a more complicated (polynomial) function.

Given this, in Figure 6 we use hierarchical signature features to compare different values N and m. The best test error is achieved using level 2 signatures. However, for all truncation levels, we observe overfitting for larger values of m. The models outperform the error bound for lower feature sizes, but the error bound decreases very rapidly to zero for larger feature sizes.

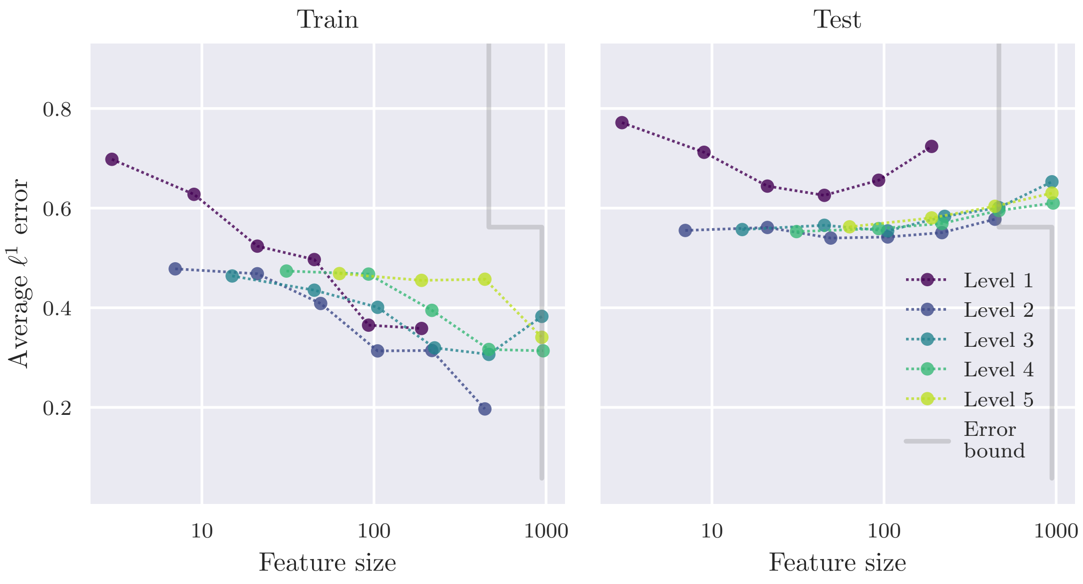

Figure 6: Train and test error, measured using ℓ^1 distance between predicted and actual solution, using hierarchical signature features. For each truncation level N, the number of levels in the hierarchical partition is increased incrementally. The error bound trace is the running minimum of the value of (24) for each point, using the number of intervals in the lowest level of the hierarchy for m.

6.3 Synthetic data: nonlinear SDE solution

We use a similar setup as above to generate a synthetic dataset, this time using a nonlinear CDE. We use the same model as [Lia+19, Example 5.1]:

$$dY_t = (-\pi Y_t + \sin(\pi t))dX_t^{(1)} + Y_t dX_t^{(2)}, \quad Y_0 = 0.$$

Taking $X_t^{(1)} = t, X_t^{(2)} = W_t$ for W_t a Brownian motion, then this is an SDE, which we consider in a Stratonovich sense. Numerically, we take a piecewise linear approximation (of length 50001) to a Brownian motion sample, and solve for the time $T = 10$ solution of the CDE using Diffrax, using a time step of $\frac{T}{50000}$ (to match [Lia+19]). We generate a dataset containing 2000 examples.

We evaluate feature sets of log signatures over an ordinary (non-hierarchical) partition for the task of predicting Y_T. We use the same kernel ridge regression framework, as described in Section 6.1. In Figure 7 we see that the most efficient feature set to achieve a given test R^2 score tends to use truncation levels 3 or 4. The test R^2 score tends to drop after a peak, as the number of subintervals increases. This may be partially due to overfitting, although we observed that the R^2 scores on the training set exhibit a similar, but less pronounced, peak. Figure 8 compares splitting by equal time, equal 1-variation, and equal 2.001-variation, with the latter being motivated by the fact that true Brownian motion sample paths almost surely have finite p-variation if and only if $p > 2$. In most cases, the equal time split was optimal.

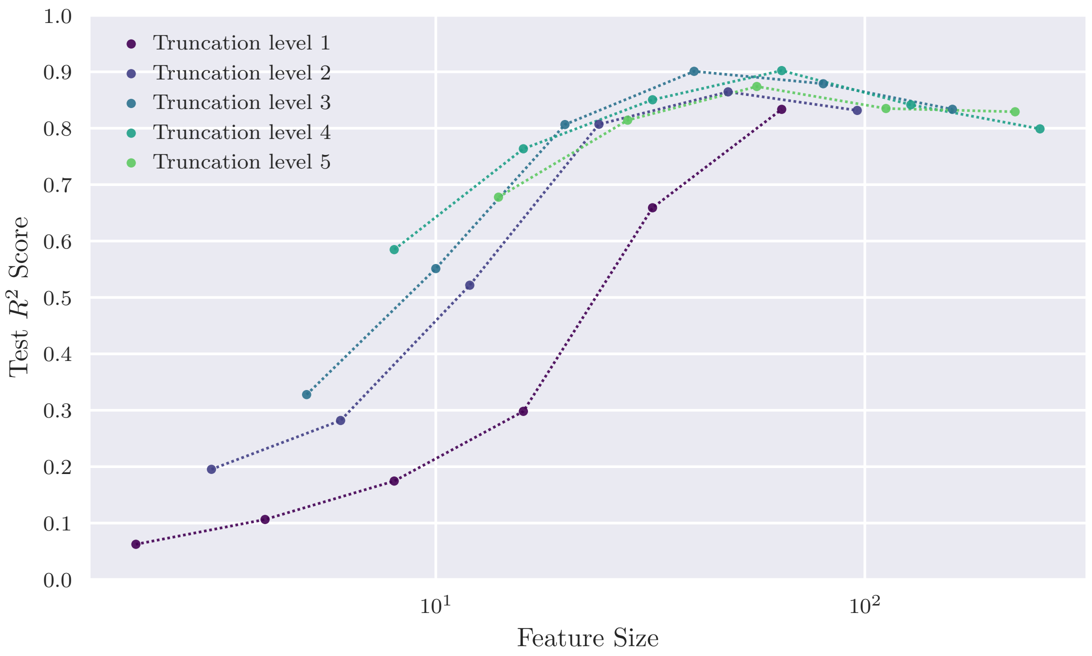

Figure 7: Test R^2 score using log signature features over subpaths (equal time split). For each truncation level, the number of subintervals ranges over $2^0, 2^1, \ldots, 2^5$. Only points with a feature size under 350 are plotted.

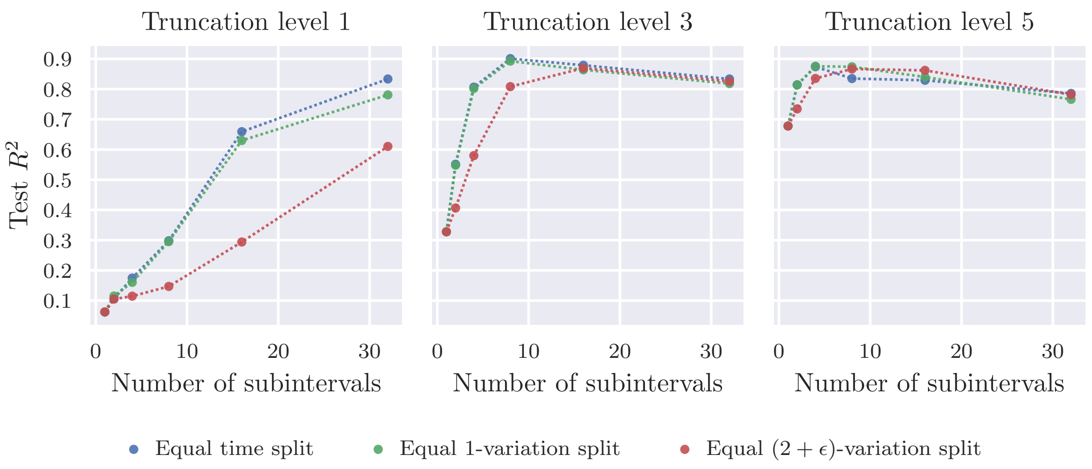

Figure 8: Test R^2 when features are created by splitting paths using an equal time split vs. equal 1-variation split vs. equal 2.001-variation split.

6.4 Detecting sleep states from accelerometer data

We now consider a dataset provided by the Child Mind Institute for a recent Kaggle competition [Esp+23]. It consists of measurements from a wrist-worn accelerometer, in a two-dimensional stream, sampled every 5 seconds over many days. The competition task was to detect the onset and end of sleep, but we instead consider the binary classification problem of predicting whether a given stream is taken from an awake period or an asleep period.

We generate a small, preprocessed dataset by extracting 800 sequences of 1440 samples (2 hours) which occur fully within a sleep or an awake period and which are at least 30 minutes away from the onset or end of sleep. Each sequence is associated with a binary label corresponding to asleep/awake.

We consider feature sets of (non-hierarchical) log signatures over subpaths, with either an equal time split or an equal 1-variation split. Since this is a classification task, we use support vector machines, with a polynomial kernel, and the framework described in Section 6.1.

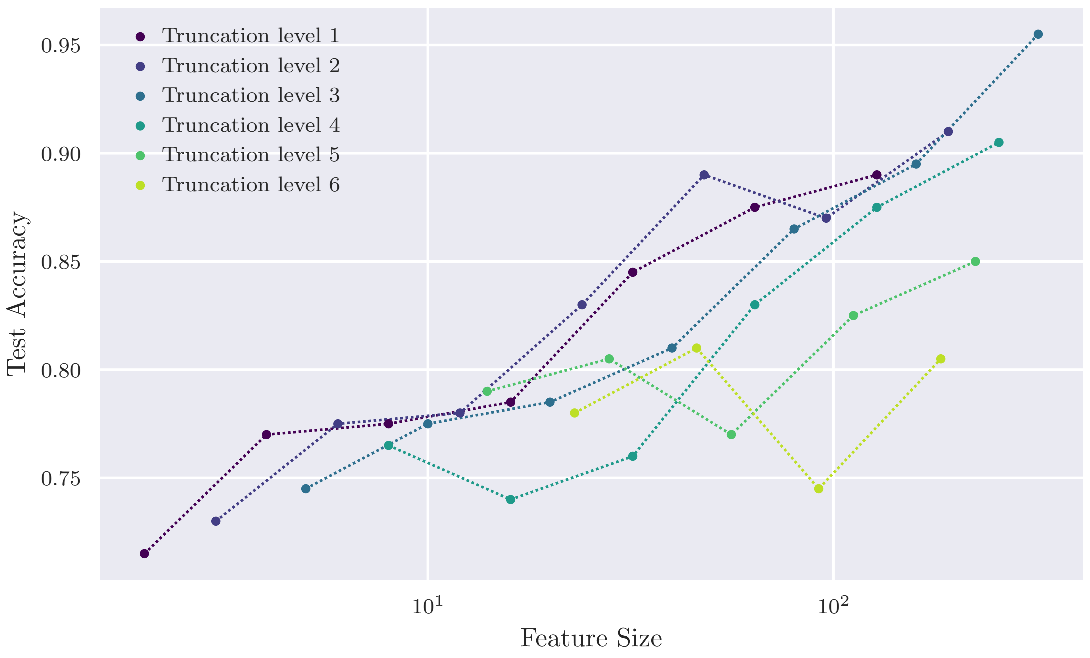

Figure 9: Test accuracy (higher is better) using log signature features over subpaths of equal 1-variation. For each truncation level, the number of subintervals ranges from $2^0, 2^1, \ldots, 2^6$, but only points with a total feature size under 350 are plotted.

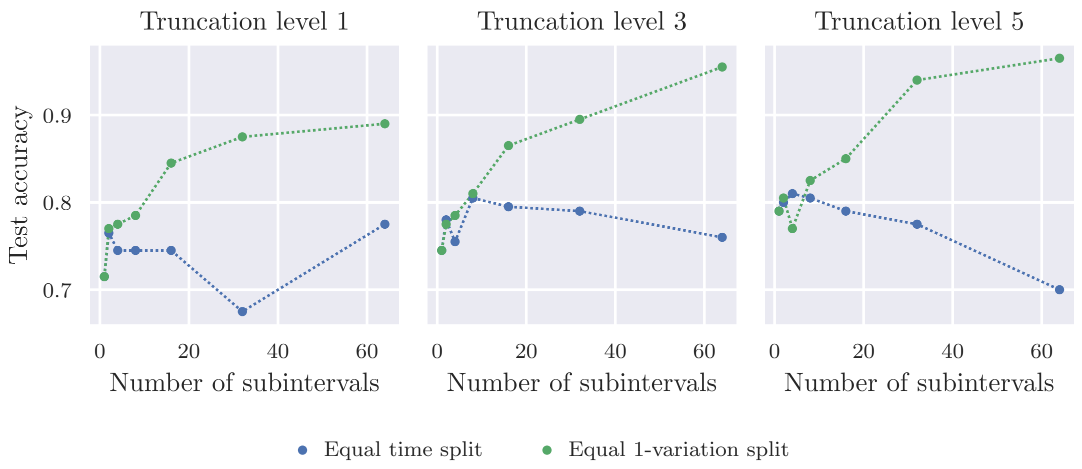

Figure 10: Comparison of test accuracy when features are created by splitting paths using an equal time split vs. equal 1-variation split.

In Figure 9 we see that the most efficient feature set achieving a given test accuracy tends to use low order log signatures (level 1 or 2). In this dataset, the paths contain occasional large "jumps", which results in the partitions being very different depending on whether the split is by equal time or equal 1-variation. Figure 10 shows the equal 1-variation split performs significantly better.

7 Conclusions

In our analysis of error bounds for linear CDE approximation, we found that the optimal trade-off between truncation level and number of subintervals depends on the dimension d and the product rL of upper bounds on the vector field norm and path length. The main trend is that for larger dimensions d, a given error bound tends to achieved most efficiently using lower order signatures (e.g. levels 2, 3, 4) over a number of subintervals.

We saw that the optimal way to choose the partition in order to minimise the error bound was to split the path into sections of equal 1-variation. On the sleep state dataset, splitting by equal 1-variation rather than equal time resulted in a significant increase in test accuracy (Figure 10). For practitioners of the path signature methodology, we highlight that in addition to being theoretically motivated, this modification is easy and efficient to implement.

Our theoretical analysis has some limitations. We primarily assumed that the paths have finite 1-variation. Also, the error bound analysis ignores the fact that in practice, the function of the feature set needs to be learned from a finite dataset. This is highlighted by the experiment on synthetic data of linear CDE solution, which shows both training and test performance decreasing as the number of subintervals increases, despite these larger feature sets containing more information (Figure 5). Future work might address this using tools from statistical learning theory.

References

[BPS23] Christian Bayer, Luca Pelizzari, and John Schoenmakers. “Primal and dual optimal stopping with signatures”. In: *arXiv preprint arXiv:2312.03444* (2023).

[Bon+19] Patric Bonnier et al. “Deep signature transforms”. In: *arXiv preprint arXiv:1905.08494* (2019).

[Che54] Kuo-Tsai Chen. “Iterated integrals and exponential homomorphisms”. In: *Proceedings of the London Mathematical Society* 3.1 (1954), pp. 502–512.

[Che57] Kuo-Tsai Chen. “Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula”. In: *Annals of Mathematics* 65.1 (1957), pp. 163–178.

[Che58] Kuo-Tsai Chen. “Integration of paths–A faithful representation of paths by noncommutative formal power series”. In: *Transactions of the American Mathematical Society* 89.2 (1958), pp. 395–407.

[CK16] Ilya Chevyrev and Andrey Kormilitzin. “A primer on the signature method in machine learning”. In: *arXiv preprint arXiv:1603.03788* (2016).

[CST23] Christa Cuchiero, Philipp Schmocker, and Josef Teichmann. “Global universal approximation of functional input maps on weighted spaces”. In: *arXiv preprint arXiv:2306.03303* (2023).

[Did] Did. *Asymptotic behavior of* $\sum_{k=1}^{n} \frac{2^k}{k}$. Mathematics Stack Exchange. Version: 28/05/2017. Accessed: 20/04/2024. URL: https://math.stackexchange.com/q/888521.

[Esp+23] Nathalia Esper et al. *Child Mind Institute - Detect Sleep States*. Accessed: 27/02/2024. 2023. URL: https://kaggle.com/competitions/child-mind-institute-detect-sleep-states.

[FH20] Peter K Friz and Martin Hairer. *A course on rough paths*. Springer, 2020.

[FV10] Peter K Friz and Nicolas B Victoir. *Multidimensional stochastic processes as rough paths: theory and applications*. Vol. 120. Cambridge University Press, 2010.

[Gyu+13] Lajos Gergely Gyurkó et al. “Extracting information from the signature of a financial data stream”. In: *arXiv preprint arXiv:1307.7244* (2013).

[HL10] Ben Hambly and Terry Lyons. “Uniqueness for the signature of a path of bounded variation and the reduced path group”. In: *Annals of Mathematics* (2010), pp. 109–167.

[Kid21] Patrick Kidger. “On neural differential equations”. PhD thesis. University of Oxford, 2021.

[KO19] Franz J Király and Harald Oberhauser. “Kernels for sequentially ordered data”. In: *Journal of Machine Learning Research* 20 (2019).

[LLN13] Daniel Levin, Terry Lyons, and Hao Ni. “Learning from the past, predicting the statistics for the future, learning an evolving system”. In: *arXiv preprint arXiv:1309.0260* (2013).

[Lia+19] Shujian Liao et al. “Learning stochastic differential equations using RNN with log signature features”. In: *arXiv preprint arXiv:1908.08286* (2019).

[Lia+21] Shujian Liao et al. “Logsig-RNN: A novel network for robust and efficient skeleton-based action recognition”. In: *arXiv preprint arXiv:2110.13008* (2021).

[LM22] Terry Lyons and Andrew D McLeod. “Signature methods in machine learning”. In: *arXiv preprint arXiv:2206.14674* (2022).

[LNO14] 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*. 2014, pp. 1–8.

[Lyo98] Terry J Lyons. “Differential equations driven by rough signals”. In: *Revista Matemática Iberoamericana* 14.2 (1998), pp. 215–310.

[LCL07] Terry J Lyons, Michael Caruana, and Thierry Lévy. *Differential equations driven by rough paths*. Springer, 2007.

[ML24] Sam Morley and Terry Lyons. *RoughPy (software library)*. https://github.com/datasig-ac-uk/RoughPy. 2024.

[Mor+19] James Morrill et al. "The signature-based model for early detection of sepsis from electronic health records in the intensive care unit". In: *2019 Computing in Cardiology (CinC)*. IEEE. 2019, Page–1.

[Mor+20] James Morrill et al. "A generalised signature method for multivariate time series feature extraction". In: *arXiv preprint arXiv:2006.00873* (2020).

[Mor+21] James Morrill et al. "Neural rough differential equations for long time series". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 7829–7838.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Rei17] Jeremy Reizenstein. "Calculation of iterated-integral signatures and log signatures". In: *arXiv preprint arXiv:1712.02757* (2017).

[Reu93] Christophe Reutenauer. *Free Lie Algebras*. LMS monographs. Clarendon Press, 1993. ISBN: 9780198536796.

[Sal+21] Cristopher Salvi et al. "The signature kernel is the solution of a Goursat PDE". In: *SIAM Journal on Mathematics of Data Science* 3.3 (2021), pp. 873–899.

[Sto48] Marshall H Stone. "The generalized Weierstrass approximation theorem". In: *Mathematics Magazine* 21.5 (1948), pp. 237–254.

[Wal+24] Benjamin Walker et al. "Log neural controlled differential equations: the Lie brackets make a difference". In: *arXiv preprint arXiv:2402.18512* (2024).

[Wit37] Ernst Witt. "Treue Darstellung Liescher Ringe." In: *Journal für die reine und angewandte Mathematik* 177 (1937), pp. 152–160. URL: http://eudml.org/doc/150011.

[YJL15] Weixin Yang, Lianwen Jin, and Manfei Liu. "Character-level Chinese writer identification using path signature feature, dropstroke and deep CNN". In: *arXiv preprint arXiv:1505.04922* (2015).

[Yan+16] Weixin Yang et al. "Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. 2016, pp. 4083–4088.

[Yan+22] Weixin Yang et al. "Developing the path signature methodology and its application to landmark-based human action recognition". In: *Stochastic Analysis, Filtering, and Stochastic Optimization: A Commemorative Volume to Honor Mark HA Davis's Contributions*. Springer, 2022, pp. 431–464.

Appendices

A Selected code excerpts

Efficient computation of the error bound (24)

```python
import math
from functools import cache

@cache
def multinomial(ks):
    # Written by Reiner Martin:
    # https://stackoverflow.com/a/46378809/5791276
    res, i = 1, sum(ks)
    i0 = ks.index(max(ks))
    for a in ks[:i0] + ks[i0+1:]:
        for j in range(1,a+1):
            res *= i
            res //= j
            i -= 1
    return res

@cache
def exponential_tail(x, k):
    """Tail of series for e^x starting from k-th term."""
    if k == 0:
        return math.exp(x)
    else:
        return exponential_tail(x, k - 1) - (x ** (k - 1)) / math.factorial(k - 1)

@cache
def D(N, k, m):
    """Computes sum over multinomial coefficients, for
    n_1 + ... + n_m = N + k, n_j <= N for all j"""
    if m == 1:
        if N + k <= N:
            return 1
        else:
            return 0
    total = 0
    for n1 in range(min(N, N + k) + 1):
        total += multinomial((n1, N + k - n1)) * D(N, k - n1, m - 1)
    return total

def C(N, k, m):
    """Sum over multinomial coefficients, for
    n_1 + ... + n_m = N + k, n_j > N for some j"""
    return m ** (N + k) - D(N, k, m)

@cache
def error_bound(N, m, rL):
    total = 0
    for k in range(1, (N * m) - N + 1):
        total += (((rL / m) ** (N + k)) * C(N, k, m) / math.factorial(N + k))
    total += exponential_tail(rL, N * m + 1)
    return total
```

Synthetic data generation using Diffrax

This example shows how the dataset for Section 6.2 is generated. The dataset for Section 6.3 is similar.

```python
import numpy as np
import jax.random as jr
import jax.numpy as jnp
from jax import vmap
import diffrax

def generate_vector_field(d, e):
    arr = np.random.randn(e, d, e)
    for i in range(d):
        # Normalise so that arr[:,i,:] has operator norm 1 wrt 1-norm
        # We do this by scaling column of the matrix arr[:,i,:] to have 1-norm equal to 1.
        col_sums = np.sum(np.abs(arr[:,i,:]), axis=0)
        arr[:,i,:] /= col_sums
    return arr

d = 2
e = 2
t0 = 0
t1 = 1
y0 = jnp.array([0.5, 0.5])
V = generate_vector_field(d, e)
diffusion = lambda t, y, args: jnp.dot(V, y)
path_n_samples = 128
path_length = 5
solver_dt0 = (t1 - t0) / 100000
path_step_size = path_length / (path_n_samples - 1)

def get_x(prng_key):
    key = jr.PRNGKey(prng_key)
    possible_increments = jnp.concatenate([jnp.eye(d), -jnp.eye(d)], axis=0)
    steps = path_step_size * jr.choice(key, possible_increments, shape=(path_n_samples - 1,))
    path = jnp.cumsum(steps, axis=0)
    path = jnp.concatenate((jnp.zeros((1,d)), path), axis=0)
    return path

def get_y(prng_key):
    path = get_x(prng_key)
    ts = jnp.linspace(t0, t1, path_n_samples)
    path_interp = diffrax.LinearInterpolation(ts=ts, ys=path)
    term = diffrax.ControlTerm(diffusion, path_interp)
    solver = diffrax.Tsit5()
    saveat = diffrax.SaveAt(t1=True)
    max_steps = int((t1 - t0) / solver_dt0) + 10
    sol = diffrax.diffeqsolve(term, solver, t0, t1, dt0=solver_dt0, max_steps=
        max_steps, y0=y0, saveat=saveat)
    y1 = sol.ys
    return y1[0]

get_xs = vmap(get_x)
get_ys = vmap(get_y)

# Generate dataset:
n_examples = 2000
X = get_xs(jnp.array(range(n_examples)))  # X has shape (2000, 128, 2)
y = get_ys(jnp.array(range(n_examples)))  # y has shape (2000, 2)
```

Generating log signature features

```

import numpy as np
import roughpy as rp

def logsig_features(path, truncation_level, n_subintervals):
    """Get log signature features with equal time split"""
    features = []
    for part in np.array_split(path, n_subintervals):
        ctx = rp.get_context(width=part.shape[1], depth=truncation_level, coeffs=rp
            .DPReal)
        stream = rp.LieIncrementStream.from_increments(np.diff(part, axis=0), ctx=
            ctx)
        logsig = stream.log_signature(depth=truncation_level)
        features.append(np.array(logsig))
    return np.array(features)

 def p_var(path, p=1):
    return np.power(np.sum(np.power(np.linalg.norm(np.diff(path, axis=0), axis=1),
        p)), 1/p)

def cumulative_p_var(path, p=1):
    return np.power(np.cumsum(np.power(np.linalg.norm(np.diff(path, axis=0), axis
        =1), p)), 1/p)

def p_var_logsig_features(path, truncation_level, n_subintervals, p):
    """Get log signature features with equal p-variation split"""
    assert p >= 1
    features = []
    cum_p_var = cumulative_p_var(path, p=p)
    num_points = path.shape[0]
    max_variation = cum_p_var[-1]
    linspace_points = np.linspace(0, max_variation, n_subintervals + 1)
    split_idxs = [np.abs(cum_p_var - point).argmin() for point in linspace_points]
    split_idxs[0] = 0
    split_idxs[-1] = num_points - 1
    for start, end in zip(split_idxs[:-1], split_idxs[1:]):
        if end - start <= 1:
            # Workaround for when end == start, which can occur
            # when there are large jumps between consecutive points.
            end = start + 1
        ctx = rp.get_context(width=path.shape[1], depth=truncation_level, coeffs=rp.DPReal)
        stream = rp.LieIncrementStream.from_increments(np.diff(path[start:end+1], axis=0), ctx=ctx)
        logsig = stream.log_signature(depth=truncation_level)
        features.append(np.array(logsig))
    return np.array(features)

 # Code for generating signature (rather than log signature) features is similar.
```

Linear CDE experiment

Example code illustrating our experimental framework.

```
from functools import partial
import itertools
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.kernel_ridge import KernelRidge
from sklearn.metrics import r2_score, mean_absolute_error

def mean_l1_distance(y_true, y_pred):
    return mean_absolute_error(y_true, y_pred, multioutput="raw_values").sum()

def eval_params(params, model, X, y, scoring):
    model = model(**params)
    cv_scores = cross_val_score(model, X, y, cv=5, scoring=scoring)
    cv_avg_score = cv_scores.mean()
    return cv_avg_score

def eval_kernel_ridge(X, y, hyperparams, test_size=0.25):
    # Mean-centre the features and targets:
    X = StandardScaler().fit_transform(X)
    y = StandardScaler().fit_transform(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=42)
    # Hyperparameter search (cross-validation)
    param_combinations = [dict(zip(hyperparams.keys(), values)) for values in itertools.product(*hyperparams.values())]
    eval_params_cv = partial(eval_params, X=X_train, y=y_train, model=KernelRidge, scoring="neg_mean_absolute_error")
    r2_scores = np.array([eval_params_cv(params) for params in param_combinations])
    best_idx = np.argmax(r2_scores)
    best_params = param_combinations[best_idx]
    # Fit model on the entire training set
    klr = KernelRidge(**best_params)
    klr.fit(X_train, y_train)
    y_pred_train = klr.predict(X_train)
    train_r2 = r2_score(y_train, y_pred_train)
    train_mean_l1 = mean_l1_distance(y_train, y_pred_train)
    # Predict on the test set and evaluate
    y_pred_test = klr.predict(X_test)
    test_r2 = r2_score(y_test, y_pred_test)
    test_mean_l1 = mean_l1_distance(y_test, y_pred_test)
    return train_r2, test_r2, train_mean_l1, test_mean_l1, best_params

def run_experiment(feature_fn, y, candidate_truncation_levels,
    candidate_n_intervals, hyperparams):
    results = pd.DataFrame(
        columns=[
            "n_subintervals",
            "truncation_level",
            "feature_size",
            "train_r2",
            "test_r2",
            "train_mean_l1_distance",
            "test_mean_l1_distance",
        ]
        + [f"optimal_{param}" for param in hyperparams.keys()]
    )
    for n_subintervals in tqdm(candidate_n_intervals, desc="n_subintervals",
        position=0):
        for truncation_level in tqdm(candidate_truncation_levels, desc="
            truncation_level", position=1, leave=False):
            X_features = feature_fn(truncation_level, n_subintervals)
            X_features = X_features.reshape(X_features.shape[0], -1)
            feature_size = X_features.shape[-1]
            train_r2, test_r2, train_mean_l1, test_mean_l1, best_params =
                eval_kernel_ridge(X_features, y, hyperparams)
            results.loc[len(results)] = [
                n_subintervals,
                truncation_level,
                feature_size,
                round(train_r2, 4),
                round(test_r2, 4),
                round(train_mean_l1, 4),
                round(test_mean_l1, 4),
            ] + [best_params.get(param) for param in hyperparams.keys()]
    return results

# Example experiment:
candidate_n_intervals = [1, 2, 4, 8, 16, 32]
candidate_truncaation_levels = [1, 2, 3, 4, 5]
hyperparams = {
    "kernel": ["poly"],
    "degree": [1, 2, 3, 4, 5],
    "alpha": [0.3, 1, 3, 10, 30],
}
results = run_experiment(
    get_logsig_features,
    # get_logsig_features is a function (omitted) which generates
    # features by stacking log signatures over subintervals
    # (can be replaced with e.g. hierarchical signatures)
    y,
    candidate_truncation_levels,
    candidate_n_intervals, hyperparams
)
```