SigGPDE: Scaling Sparse Gaussian Processes on Sequential Data

Maud Lemercier [1] **Cristopher Salvi** [2] **Thomas Cass** [3] **Edwin V. Bonilla** [4] **Theodoros Damoulas** [1] **Terry Lyons** [2]

Abstract

Making predictions and quantifying their uncertainty when the input data is sequential is a fundamental learning challenge, recently attracting increasing attention. We develop SigGPDE, a new scalable sparse variational inference framework for Gaussian Processes (GPs) on sequential data. Our contribution is twofold. First, we construct inducing variables underpinning the sparse approximation so that the resulting evidence lower bound (ELBO) does not require any matrix inversion. Second, we show that the gradients of the GP signature kernel are solutions of a hyperbolic partial differential equation (PDE). This theoretical insight allows us to build an efficient back-propagation algorithm to optimize the ELBO. We showcase the significant computational gains of SigGPDE compared to existing methods, while achieving state-of-the-art performance for classification tasks on large datasets of up to 1 million multivariate time series.

1. Introduction

Gaussian process (GP) models provide a sound mathematical framework for supervised learning that allows the incorporation of prior assumptions and provides uncertainty estimates when modelling unknown functions (Rasmussen & Williams, 2006). This is usually achieved by specifying a GP prior over functions with a suitable covariance (or kernel) along with a conditional likelihood. With this, the problem boils down to that of estimating the posterior over the function (values) given the observed data.

However, this posterior distribution is often analytically intractable and, even when the conditional likelihood is a Gaussian, GP models scale poorly on the number of observations N, with naïve approaches having a time complexity $\mathcal{O}(N^3)$. From a wide range of approximate techniques to scale inference in GP models to large datasets, “sparse” methods based on variational inference (VI) have emerged as one of the dominant approaches (Titsias, 2009). They consist in defining a family of approximate posteriors through M *inducing variables*, and selecting the distribution in this family that minimizes the Kullback-Leibler (KL) divergence between the approximation and the true posterior. This is achieved by minimizing the so-called evidence lower bound (ELBO). When the likelihood factorizes over datapoints, training can be done in minibatches of size $\tilde{N}$ resulting in a per-iteration computational cost $\mathcal{O}(\tilde{N}M^2 + M^3)$, where the $\mathcal{O}(M^3)$ cost is due to the inversion of the covariance matrix of the M inducing variables. This yields significant computational savings when $M \ll N$.

In the seminal work of Titsias (2009) the inducing variables correspond to evaluations of the GP at M pseudo input locations, which typically results in a dense covariance matrix to invert. Subsequently, other ways of constructing inducing variables have been introduced in order to mitigate the $\mathcal{O}(M^3)$ cost (Hensman et al., 2017; Burt et al., 2020b). The core idea consists in defining (almost) independent inducing variables, such that their covariance matrix is (almost) diagonal. These inducing variables correspond to projections of the GP on basis functions, such that the covariance matrix is a Gramian matrix with respect to some inner-product. Orthogonal basis functions yield diagonal Gramian matrices, hence these methods are often referred to as *variational orthogonal features* (VOFs) . However existing VOF methods are limited to stationary kernels on $\mathcal{X} \subset \mathbb{R}^d$ ($d \in \mathbb{N}$).

In this work we are interested in generalizing the VOF paradigm to the case where the input space $\mathcal{X}$ is a set of *sequences* of vectors in $\mathbb{R}^d$. One may be tempted to naively concatenate each vector in a sequence of length ℓ to form a flat vector in $\mathbb{R}^{\ell d}$. However in this case existing VOF methods cannot be directly applied because they are limited to low dimensional vectors, with $d \leq 8$ (Dutordoir et al., 2020). Thus, one needs kernel functions specifically designed for sequential data. The *signature kernel* (Cass et al., 2020) is a natural choice that has recently emerged as a leading machine learning tool for learning on sequential data. In particular, Toth & Oberhauser (2020) have proposed a GP inference framework leveraging an approximation of this covariance function (Király & Oberhauser, 2019) and achieving state-of-the-art performance on time series classi-

[1]University of Warwick and Alan Turing Institute [2]University of Oxford and Alan Turing Institute [3]Imperial College London and Alan Turing Institute [4]CSIRO’s Data61. Correspondence to: Maud Lemercier <maud.lemercier@warwick.ac.uk>.

fication tasks. Nevertheless, as in standard sparse variational approaches to GPs, the inducing inputs they chose (so called *inducing tensors*) are additional variational parameters to optimize, and the resulting covariance matrix is dense.

Here we develop SigGPDE, a new scalable sparse variational inference framework for GP models on sequential data. After a brief recap on the general principles of variation inference (Sec. 2) we identify a set of VOFs *naturally* associated with the signature kernel. These inducing variables do not depend on any variational parameter as they are defined as projections of GP-samples onto an orthogonal basis for the RKHS associated to the signature kernel (Sec. 3). As a result, unlike the methods developed in Toth & Oberhauser (2020), in SigGPDE the optimization of the ELBO *does not require any matrix inversion*. Subsequently, we show that the gradients of the signature kernel are solutions of a *hyperbolic partial differential equation* (PDE). This theoretical insight allows us to build an efficient backpropagation algorithm to optimize the ELBO (Sec. 4). Our experimental evaluation shows that SigGPDE is considerably faster than GPSig, whilst retaining similar predictive performances on datasets of up to 1 million multivariate time series (Sec. 6).

2. Background

We begin with a general summary of variational inference for GPs. In this section, it is assumed that the input space $\mathcal{X} \subset \mathbb{R}^d$. Standard models with zero-mean GP priors and iid conditional likelihoods can be written as follows

$$f \sim \mathcal{GP}(0, k(\cdot,\cdot)) \quad p(\mathbf{y}|\mathbf{f}) = \prod_{i=1}^{N} p(y_i|f(x_i)), \tag{1}$$

where $k(\cdot,\cdot)$ is the covariance function. The general setting for sparse GPs consists in specifying a collection of M variables as well as a joint distribution with variational parameters $\mathbf{m}$ (mean vector) and Σ (covariance matrix)

$$\mathbf{u} = \{u_m\}_{m=1}^{M}, \quad q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \Sigma). \tag{2}$$

These variables induce a family of approximate posteriors that are GPs with finite dimensional marginal densities of the form $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$. Considering any input $x \in \mathcal{X}$, the mean and covariance functions of these GPs are

$$\mu_q(x) = C_{f_x\mathbf{u}} C_{\mathbf{uu}}^{-1}\mathbf{m} \tag{3}$$
$$k_q(x,y) = k(x,y) - C_{f_x\mathbf{u}} C_{\mathbf{uu}}^{-1}(C_{\mathbf{uu}} - \Sigma)C_{\mathbf{uu}}^{-1}C_{\mathbf{u}f_y},$$

where the vector $C_{f_x\mathbf{u}}$ and the matrix $C_{\mathbf{uu}}$ are defined as

$$[C_{f_x\mathbf{u}}]_m = \mathbb{E}[u_m f(x)], \quad [C_{\mathbf{uu}}]_{m,m'} = \mathbb{E}[u_m u_{m'}] \tag{4}$$

Provided the inducing variables $\mathbf{u}$ are deterministic conditioned on f, one has the following lower bound (ELBO) on the marginal log likelihood (Matthews, 2017)

$$\log p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f})] - KL[q(\mathbf{u})||p(\mathbf{u})], \tag{5}$$

where $p(\mathbf{u}) = \mathcal{N}(0_M, C_{\mathbf{uu}})$. Maximizing the right-hand-side of eq. (5) is equivalent to minimizing the KL divergence between $q(f)$ and the true posterior distribution.

The original variational inference framework outlined in Titsias (2009) consists in setting $u_m = f(z_m)$ where $z_m \in \mathcal{X}$ is a pseudo input living in the same space as x that may either be fixed or optimized. The per-iteration cost of optimizing the ELBO is $\mathcal{O}(\tilde{N}M^2 + M^3)$, where $\tilde{N}$ is the minibatch size, and M^3 is the cost of computing $C_{\mathbf{uu}}^{-1}$. Therefore, the computational bottleneck of sparse GPs is in the inversion of the matrix $C_{\mathbf{uu}}$ via a Cholesky decomposition that has a complexity $\mathcal{O}(M^3)$.

Recently, a considerable effort has been devoted to the construction of inducing variables $\mathbf{u}$ which yield a structured covariance matrix $C_{\mathbf{uu}}$ whose inversion has a reduced computational complexity (Hensman et al., 2017). This line of work is often referred to as *inter-domain* sparse GPs, owing to the fact that the pseudo inputs are not constrained to live in $\mathcal{X}$ as before. In particular, Burt et al. (2020b); Dutordoir et al. (2020) have shown that provided one can find an orthogonal basis of functions for the RKHS associated with the kernel $k(\cdot,\cdot)$, it is possible to define the inducing variables as projections of the GP samples onto this basis. This construction yields a diagonal covariance matrix $C_{\mathbf{uu}}$.

3. Variational Inference with Orthogonal Signature Features

Here we present our first contribution, namely the use of *orthogonal signature features* as inducing variables for GPs on sequential data. We begin with a summary of the theoretical background needed to define GPs endowed with the signature kernel. In this section $\mathcal{X}$ is no longer a subspace of $\mathbb{R}^d$ but will be defined as a space of *paths* hereafter.

3.1. The signature

Consider a time series $\mathbf{x}$ as a collection of points $x_i \in \mathbb{R}^{d-1}$ with corresponding time-stamps $t_i \in \mathbb{R}$ such that

$$\mathbf{x} = ((t_0, x_0), (t_1, x_1), ..., (t_n, x_n)) \tag{6}$$

with $0 = t_0 < ... < t_n = T$. Let $X : [0,T] \to \mathbb{R}^d$ be the piecewise linear interpolation of the data such that $X_{t_i} = (t_i, x_i)$. We denote by $\mathcal{X}$ the set of all continuous piecewise linear paths defined over the time interval $[0,T]$ and with values on $\mathbb{R}^d$.

For any path $X \in \mathcal{X}$ and any $\alpha \in \{1,\ldots,d\}$, we will denote its α^{th} *channel* by $X^{(\alpha)}$ so that at any time $t \in [0,T]$

$$X_t = (X_t^{(1)}, \ldots, X_t^{(d)}) \tag{7}$$

The *signature* $S : \mathcal{X} \to H$ is a *feature map* defined for any path $X \in \mathcal{X}$ as the following infinite collection of statistics

$$S(X) = \Big(1, \Big\{S(X)^{(\alpha_1)}\Big\}_{\alpha_1=1}^{d}, \Big\{S(X)^{(\alpha_1,\alpha_2)}\Big\}_{\alpha_1,\alpha_2=1}^{d}, \Big\{S(X)^{(\alpha_1,\alpha_2,\alpha_3)}\Big\}_{\alpha_1,\alpha_2,\alpha_3=1}^{d}, \dots\Big)$$

where each term is a scalar equal to the iterated integral

$$S(X)^{(\alpha_1,\dots,\alpha_j)} = \underset{0<s_1<\dots<s_j<T}{\int \dots \int} dX_{s_1}^{(\alpha_1)} \dots dX_{s_j}^{(\alpha_j)} \quad (8)$$

The *feature space* H associated to the signature is a Hilbert space defined as the direct sum of tensor powers of $\mathbb{R}^d$

$$H = \bigoplus_{k=0}^{\infty} (\mathbb{R}^d)^{\otimes k} = \mathbb{R} \oplus \mathbb{R}^d \oplus (\mathbb{R}^d)^{\otimes 2} \oplus \dots \quad (9)$$

where $\otimes$ denotes the outer product (Lyons, 1998; 2014).

Interpretability of the signature features When dealing with signals with multiple channels $X^{(1)}, \dots, X^{(d)}$, cause-effect relations between the different channels might be an essential feature that one wishes to extract from the signal. Intrinsic in the definition of the signature are the iterated integrals of eq. (8). The order $s_1 < \dots < s_j$ of the domain of integration, naturally captures interactions between the channels and provides the signature features with a natural interpretability (Moore et al., 2019; Lemercier et al., 2021).

3.2. The signature kernel

The *signature kernel* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a reproducing kernel associated to the signature feature map and defined for any pair of paths $X, Y \in \mathcal{X}$ as the following inner product

$$k(X,Y) = \langle S(X), S(Y) \rangle_H \quad (10)$$

It their recent article, Cass et al. (2020) provide a *kernel trick* for the signature kernel by proving the relation

$$k(X,Y) = U(T,T) \quad (11)$$

where the function of two variables $U : [0,T] \times [0,T] \to \mathbb{R}$ is the solution of the following hyperbolic PDE

$$\frac{\partial^2 U}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t) U \quad (12)$$

with boundary conditions $U(0,\cdot) = 1$ and $U(\cdot,0) = 1$.

From the structure of H and the properties of the signature it turns out that the signature kernel can be decomposed according to the expansion (Cass et al., 2020)

$$k(X,Y) = \sum_{j=0}^{\infty} \sum_{|\boldsymbol{\alpha}|=j} S(X)^{\boldsymbol{\alpha}} S(Y)^{\boldsymbol{\alpha}}. \quad (13)$$

where the inner summation is over the set of multi-indices

$$\{\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_j) : \alpha_1, \dots, \alpha_j \in \{1, \dots, d\}\} \quad (14)$$

Next, we propose a simple parametrization of this kernel.

3.3. Parametrization of the signature kernel

In many real-world problems the input path X contains a large number d of different channels, only some of which are relevant. For any coordinate $\alpha \in \{1, \dots, d\}$ and time index $t \in [0,T]$ one can rescale each channel $X^{(\alpha)}$ by a scalar hyperparameter θ_α yielding the rescaled path

$$X_t^{\boldsymbol{\theta}} := (\theta_1 X_t^{(1)}, \dots, \theta_d X_t^{(d)}). \quad (15)$$

From eq. (8) it is straightforward to see that the corresponding rescaled signature satisfies the following relation

$$S_{\boldsymbol{\theta}}(X)^{(\alpha_1,\dots,\alpha_j)} := S(X^{\boldsymbol{\theta}})^{(\alpha_1,\dots,\alpha_j)} \quad (16)$$

$$= \theta_{\alpha_1} \dots \theta_{\alpha_j} S(X)^{(\alpha_1,\dots,\alpha_j)} \quad (17)$$

for any $\alpha_1, \dots, \alpha_j \in \{1, \dots, d\}$. As a result, akin to an *automatic relevance determination* (ARD) parametrization, the signature kernel of eq. (13) can be reparametrized as

$$k_{\boldsymbol{\theta}}(X,Y) = \sum_{j=0}^{\infty} \sum_{|\boldsymbol{\alpha}|=j} S_{\boldsymbol{\theta}}(X)^{\boldsymbol{\alpha}} S_{\boldsymbol{\theta}}(Y)^{\boldsymbol{\alpha}}. \quad (18)$$

3.4. Variational Orthogonal Signature Features

The *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}$ associated to the signature kernel $k_{\boldsymbol{\theta}}$ can be defined as

$$\mathcal{H} = \{g : X \mapsto \langle S_{\boldsymbol{\theta}}(X), \mathbf{h} \rangle_H\}, \quad \mathbf{h} \in H. \quad (19)$$

For any two functions $g_1, g_2 \in \mathcal{H}$ such that

$$g_1 : X \mapsto \langle S_{\boldsymbol{\theta}}(X), \mathbf{h}_1 \rangle_H \quad (20)$$

$$g_2 : X \mapsto \langle S_{\boldsymbol{\theta}}(X), \mathbf{h}_2 \rangle_H, \quad (21)$$

the inner product $\langle \cdot, \cdot \rangle_H$ induces the inner product on $\mathcal{H}$

$$\langle g_1, g_2 \rangle_{\mathcal{H}} = \langle \mathbf{h}_1, \mathbf{h}_2 \rangle_H. \quad (22)$$

The key to our setup is that the set of signature features

$$\mathcal{S}^{\perp} = \{S_{\boldsymbol{\theta}}(\cdot)^{\boldsymbol{\alpha}} : X \mapsto S_{\boldsymbol{\theta}}(X)^{\boldsymbol{\alpha}}\}_{\boldsymbol{\alpha}=(\alpha_1,\dots,\alpha_j)} \quad (23)$$

form an orthonormal basis for the RKHS $\mathcal{H}$, i.e.

$$\Big\langle S_{\boldsymbol{\theta}}(\cdot)^{\boldsymbol{\alpha}}, S_{\boldsymbol{\theta}}(\cdot)^{\boldsymbol{\alpha}'} \Big\rangle_{\mathcal{H}} = \delta_{\boldsymbol{\alpha},\boldsymbol{\alpha}'} = \begin{cases} 1, & \text{if } \boldsymbol{\alpha} = \boldsymbol{\alpha}', \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

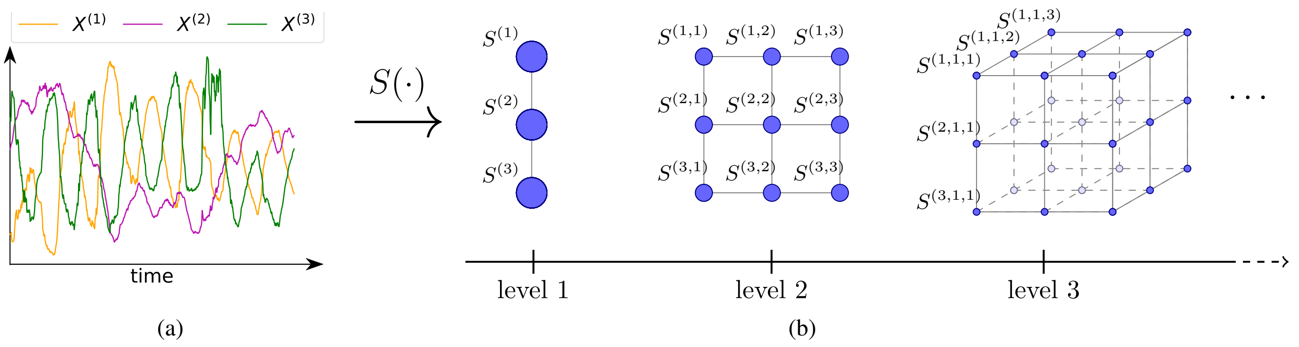
$X^{(1)}$
$X^{(2)}$
$X^{(3)}$
time
$S(\cdot)$
$S^{(1)}$ $S^{(2)}$ $S^{(3)}$
$S^{(1,1)}$ $S^{(1,2)}$ $S^{(1,3)}$ $S^{(2,1)}$ $S^{(2,2)}$ $S^{(2,3)}$ $S^{(3,1)}$ $S^{(3,2)}$ $S^{(3,3)}$
$S^{(1,1,3)}$ $S^{(1,1,2)}$ $S^{(1,1,1)}$ $S^{(2,1,1)}$ $S^{(3,1,1)}$
level 1
level 2
level 3
(a)
(b)

Figure 1. Illustration of the first terms of the signature $S(X)$ for a 3-dimensional path X. Each blue circle corresponds to a signature feature $S(X)^{\boldsymbol{\alpha}}$ with $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_j)$. The size of the circle reflects the feature importance according to the property $|S(X)^{\boldsymbol{\alpha}}| = \mathcal{O}(1/|\boldsymbol{\alpha}|!)$. The first feature $S^{(0)}$ which is always equal to 1 is omitted in this schematic.

An important property of the orthonormal basis $\mathcal{S}^{\perp}$ is that its elements are naturally ordered[1] $S^1_{\boldsymbol{\theta}}, S^2_{\boldsymbol{\theta}}, \dots, S^m_{\boldsymbol{\theta}}, \dots$ This ordering is due to the property that for any path $X \in \mathcal{X}$ the terms of the signature decay factorially (Lyons et al., 2007)

$$|S(X)^{\boldsymbol{\alpha}}| = \mathcal{O}\left(\frac{1}{|\boldsymbol{\alpha}|!}\right), \tag{25}$$

as shown in Fig. 1. From eqs. (24) and (25) we define our inducing variables as orthogonal projections of the GP[2] onto the first M elements of the orthonormal signature basis $\mathcal{S}^{\perp}$

$$u_m = \langle f, S^m_{\boldsymbol{\theta}} \rangle_{\mathcal{H}}, \quad 1 \leq m \leq M. \tag{26}$$

With this choice of inducing variables we easily deduce the following covariances (Hensman et al., 2017)

$$\mathbb{E}[u_m f(X)] = S^m_{\boldsymbol{\theta}}(X) \quad \text{and} \quad \mathbb{E}[u_m u_{m'}] = \delta_{m,m'}. \tag{27}$$

This implies that the covariance matrix $C_{\mathbf{uu}}$ is the identity.

For any path $X \in \mathcal{X}$ we use the convenient vector notation

$$\mathcal{S}_M(X) := [S^1_{\boldsymbol{\theta}}(X), \dots, S^M_{\boldsymbol{\theta}}(X)] \in \mathbb{R}^M \tag{28}$$

to obtain the approximate posterior $\mathcal{GP}(\mu, \nu)$ with mean and covariance functions defined by the following equations

$$\begin{aligned} \mu(X) &= \mathcal{S}_M(X)^T \mathbf{m} \qquad (29)\\ \nu(X,Y) &= k_{\boldsymbol{\theta}}(X,Y) - \mathcal{S}_M(X)^T (I_M - \Sigma) \mathcal{S}_M(Y). \end{aligned}$$

We note that the signature and the signature kernel can be easily computed on real time series using existing python libraries (Lyons, 2010; Reizenstein & Graham, 2018).

[1]To index the signature orthogonal features we can order them first by increasing level j, and then by sorting the multi-indices $\boldsymbol{\alpha}$.

[2]Although f does not belong to $\mathcal{H}$ (Kanagawa et al., 2018), such projections are well defined.

4. Reverse-mode automatic differentiation for the signature kernel

In order to optimize the ELBO with respect to the parameters $\boldsymbol{\theta}$ one needs to take derivatives of the signature kernel $k_{\boldsymbol{\theta}}$ of eq. (29) with respect to each of its input paths. Given that $k_{\boldsymbol{\theta}}$ solves the PDE (12) it can be computed using appropriate PDE numerical solvers. Therefore, in theory the differentiation could be carried out by leveraging the automatic differentiation tools of modern deep learning libraries (Tensorflow, PyTorch etc.). However, backpropagating through the operations of the PDE solver can be inefficient.

Here we show that the gradients of $k_{\boldsymbol{\theta}}$ can be computed efficiently *without backpropagating through the operations of the PDE solver* as they are the solutions of a second PDE analogous to eq. (12). The ability not to rely on automatic differentation allows for an efficient fitting of SigGPDE both in the terms of time complexity and memory cost.

4.1. Differentiating the signature kernel along the direction of a path

Consider a time series $\mathbf{x}$ as a collection of points $x_i \in \mathbb{R}^d$ with corresponding time-stamps $s_i \in \mathbb{R}$ such that

$$\mathbf{x} = ((s_0, x_0), (s_1, x_1), ..., (s_\ell, x_\ell)) \tag{30}$$

with $s_0 < ... < s_\ell$. Every vector x_i in the sequence can be written with respect to the canonical basis of $\mathbb{R}^d$ as

$$x_i = \sum_{j=1}^{d} x_{i,j} \mathbf{e}_j \tag{31}$$

Let $X : [0,T] \to \mathbb{R}^d$ be the piecewise linear interpolation of the data such that $X_{t_i} = (t_i, x_i)$. Similarly for a second time series $\mathbf{y}$ and resulting piecewise linear interpolation Y.

Algorithm 1 Backpropagation for $k_{\boldsymbol{\theta}}(X, X)$ via PDE (41)

```
1: Input: Path $X$, localised impulses $\boldsymbol{\gamma} = \{\gamma_{i,j}\}$ fully determined by the time series $\mathbf{x}$.
2:     $\mathbf{u}_{0,:} = [1, 0, \ldots, 0]$,  $\mathbf{u}_{:,0} = [1, 0, \ldots, 0]$        // Boundary conditions for the augmented state
3:     def aug_dynamics($[U(s,t), U_{\boldsymbol{\gamma}}(s,t)], s, t$):        // Dynamics for the augmented state
4:         return $\left[\dot{X}_s^T \dot{X}_t U(s,t), \dot{X}_s^T \dot{X}_t U_{\boldsymbol{\gamma}}(s,t) + \dot{\boldsymbol{\gamma}}_s \dot{X}_t U(s,t)\right]$
5:     $[U(T,T), U_{\boldsymbol{\gamma}}(T,T)] = \mathrm{PDESolve}(\mathbf{u}_{0,:}, \mathbf{u}_{:,0}, \mathrm{aug\_dynamic}, T, T)$
6: Output: $2 \cdot U_{\boldsymbol{\gamma}}(T,T)$        // Gradients of the kernel at the knots of $X$
```

Recall the definition of signature kernel as

$$k_{\boldsymbol{\theta}}(X, Y) = k(X^{\boldsymbol{\theta}}, Y^{\boldsymbol{\theta}}), \tag{32}$$

where $X^{\boldsymbol{\theta}}$ and $Y^{\boldsymbol{\theta}}$ are the rescaled paths of eq. (15).

By the chain rule one has that

$$\frac{\partial k_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} = \frac{\partial k}{\partial X^{\boldsymbol{\theta}}} \frac{\partial X^{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} + \frac{\partial k}{\partial X^{\boldsymbol{\theta}}} \frac{\partial Y^{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} \tag{33}$$

Hence, to formulate a backpropagation algorithm in a rigorous way compatible with the TensorFlow library used in this work, we need to give meaning to the following gradients

$$\left\{ \frac{\partial}{\partial x_{i,j}} k(X, Y) \right\}_{i,j=1}^{\ell, d} \tag{34}$$

The technical difficulty here consists in reconciling the continuous nature of the input path X and the discrete nature of the locations $x_{i,j}$ where one wants to compute the gradients and given by the knots of the time series $\mathbf{x}$.

Next we introduce a collection of *localised impulses* and define the concept of *directional derivative of the signature kernel along a path* in order to make sense of the gradients in eq. (34). These definitions will be followed by the main result of this section, namely that the directional derivative of k solves another PDE similar to eq. (12) for the signature kernel, for which we derive an explicit solution via the *technique of variation of parameters* (Thm. 4.1).

Definition 1. *For any $i = 1, \ldots, \ell$ and any $j = 1, \ldots, d$ define the localised impulse $\gamma_{i,j} : [0, T] \to \mathbb{R}^d$ as the solution of the following ordinary differential equation (ODE)*

$$\dot{\gamma}_{i,j} = \frac{1}{\ell} e_j \mathbb{1}_{\{t \in [(i-1)/\ell, i/\ell)]\}}, \quad \gamma_{i,j}(0) = 0 \tag{35}$$

Definition 2. *For any path $\gamma \in \mathcal{X}$ the directional derivative of the signature kernel k along γ is defined as*

$$k_\gamma(X, Y) := \frac{\partial}{\partial \epsilon} k\Big(X + \epsilon\gamma, Y\Big)\Big|_{\epsilon=0} \tag{36}$$

Each gradient of the signature kernel $k_{\boldsymbol{\theta}}$ at the knot $x_{i,j}$ reported in eq. (34) can be identified with the directional derivative of $k_{\boldsymbol{\theta}}$ along the localised impulse $\gamma_{i,j}$ of Def. 1

$$\frac{\partial}{\partial x_{i,j}} k(X, Y) := k_{\gamma_{i,j}}(X, Y) \tag{37}$$

4.2. A PDE for the gradients of the signature kernel

Recall that the signature kernel k_θ solves the following PDE

$$\frac{\partial^2 U}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t) U \tag{38}$$

Integrating both sides with respect to s and t one obtains

$$U(s,t) = 1 + \int_{u=0}^{s} \int_{v=0}^{t} U(u,v)(\dot{X}_u^T \dot{Y}_v) du dv \tag{39}$$

Let's denote by $U_\gamma : [0,T] \times [0,T] \to \mathbb{R}$ the directional derivative k_γ evaluated at the restricted paths $X|_{[0,s]}, Y|_{[0,t]}$

$$U_\gamma(s,t) := k_\gamma(X|_{[0,s]}, Y|_{[0,t]}) \tag{40}$$

The combination of eqs. (39) and (40) yields the relation

$$\begin{aligned} U_\gamma(s,t) &= \frac{\partial}{\partial \epsilon} k\Big((X + \epsilon\gamma)|_{[0,s]}, Y|_{[0,t]}\Big)\Big|_{\epsilon=0} \\ &= \frac{\partial}{\partial \epsilon} \left(\int_0^s \int_0^t U(u,v) \left(\dot{X}_u + \epsilon \dot{\gamma}_u\right)^T \dot{Y}_v du dv \right)_{\epsilon=0} \\ &= \int_0^s \int_0^t \left(U_\gamma(u,v) \dot{X}_u^T \dot{Y}_v + U(u,v) \dot{\gamma}_u^T \dot{Y}_v \right) du dv \end{aligned}$$

Hence, differentiating the last equation first with respect to t and then s we get that the directional derivative k_γ of the signature kernel along the path γ solves the following PDE

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t) U_\gamma + (\dot{\gamma}_s^T \dot{Y}_t) U \tag{41}$$

with boundary conditions

$$U_\gamma(0, \cdot) = 0, \quad U_\gamma(\cdot, 0) = 0. \tag{42}$$

As a result, the gradients in eq. (34) of the signature kernel with respect to each of its input paths can be computed in a single call to a PDE solver, which concatenates the original state and the partial derivatives (41) into a single vector. Each partial derivative follows the dynamics of (41) where one replaces the direction γ by the relevant localised impulse $\gamma_{i,j}$, $\tau_{i,j}$ for X and Y respectively. We outline the resulting procedure in Alg. 1, where the concatenated partial derivatives are denoted by $U_{\boldsymbol{\gamma}}(s,t)$. Note that to optimize the ELBO we only need to differentiate $k(X, X)$, which is the case presented in the algorithm. The generalization to the case $k(X, Y)$ is straightforward using the chain rule.

Algorithm 2 Backpropagation for $k_{\boldsymbol{\theta}}(X, X)$ via variation of parameters (Thm. 4.1)

```
1: Input: Path $X$, localised impulses $\boldsymbol{\gamma} = \{\gamma_{i,j}\}$ fully determined by the time series $\mathbf{x}$
2:     $\mathbf{u}_{0,:} = [1, \dots, 1]$,  $\mathbf{u}_{:,0} = [1, \dots, 1]$          // Boundary conditions for the augmented state
3:     def aug_dynamics$\left(\left[U(s,t), \tilde{U}(s,t)\right], s, t\right)$:          // Dynamics for the augmented state
4:         return $\left[\dot{X}_s^T \dot{X}_t U(s,t), \dot{X}_{T-s}^T \dot{X}_{T-t} \tilde{U}(s,t)\right]$
5:     $[U, \tilde{U}] = \text{PDESolve}(\mathbf{u}_{0,:}, \mathbf{u}_{:,0}, \text{aug\_dynamic}, T, T)$          // Keep the solutions at each $(s,t)$
6: $U_{\boldsymbol{\gamma}} = \text{tf.sum}(U \cdot \tilde{U} \cdot \boldsymbol{\gamma} X)$          // Simple final TensorFlow operations
7: Output: $2 \cdot U_{\boldsymbol{\gamma}}$          // Gradients of the kernel at the knots of $X$
```

4.3. An explicit solution by variation of parameters

From this second PDE (41) we derive the following theorem (proved in the appendix), that allows to compute the directional derivative k_γ of the signature kernel directly from its evaluations at X, Y and at $\overleftarrow{X}, \overleftarrow{Y}$, where $\overleftarrow{X}, \overleftarrow{Y}$ are respectively the paths X, Y reversed in time.

Theorem 4.1. *For any $\gamma \in \mathcal{X}$ the directional derivative $k_\gamma(X, Y)$ of the signature kernel along the path γ satisfies the following relation*

$$k_\gamma(X, Y) = \int_0^T \int_0^T U(s,t)\widetilde{U}(T-s, T-t)(\dot{\gamma}_s^T \dot{Y}_t) ds dt$$

where $\widetilde{U}(s,t) = k(\overleftarrow{X}|_{[0,s]}, \overleftarrow{Y}|_{[0,t]})$ and where $\overleftarrow{X}, \overleftarrow{Y}$ are respectively the paths X, Y reversed in time.

The full backpropagation procedure is described in Alg. 2.

5. Related work

In this section we expand on the material presented in Sec. 2, focusing on the most recent approaches to scalable GPs on $\mathbb{R}^d$ with VOFs and on sparse GPs for sequential data.

Variational Fourier Features In Hensman et al. (2017) the inducing variables are defined for scalar input $\mathcal{X} = \mathbb{R}$ as projections of the GP-sample onto the truncated Fourier basis. This type of inducing variables can be constructed for GPs with Matérn-type kernels. Although the resulting covariance matrix of the inducing variables is not diagonal, it can be decomposed into the sum of a diagonal matrix and rank one matrices. As a result it can be inverted using the *Woodbury identity*, which makes it possible to scale GP inference on $\mathbb{R}$. The generalization to GPs on $\mathbb{R}^d$ is done by taking the outer product of the Fourier basis on $\mathbb{R}$.

Eigenfunction inducing features Closest to our work is the eigenfunction inducing features developed by Burt et al. (2020a), where the inducing variables are also defined as projections of the GP-sample onto an orthogonal basis of functions for the RKHS associated with the GP kernel. This relies on a *Mercer's expansion* of the kernel. From here one identifies this orthogonal basis functions by solving an eigendecomposition problem. For example Dutordoir et al. (2020) map the input data to the hypersphere $S^{d-1} \subset \mathbb{R}^d$ and then show that *spherical harmonics* form an orthogonal basis for RKHS associated to *zonal kernels* defined on S^{d-1}.

GPs with signature covariances Toth & Oberhauser (2020) propose a different sparse GP inference framework for sequential data with signature covariances (GPSig). In this work the inducing variables are either taken to be *inducing sequences* (IS) in the original input space (GPSig-IS) of sequences or *inducing tensors* (IT) in the corresponding feature space (GPSig-IT). The chosen covariance function is an approximation of the signature kernel based on truncating the signature to a finite level n. This truncation makes the feature space finite dimensional and allows to optimize inducing tensors defined over such truncated space. Unlike our method, the inducing tensors are additional variational parameters to optimize. The covariance matrix $C_{\mathbf{uu}}$ is dense and its inversion incurs an additional $\mathcal{O}(M^3)$ cost. In Table 1 we compare the computational complexities of GPSig-IT, GPSig-IS and SigGPDE. A similar table for the memory complexity can be found in the appendix.

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$C_{\mathbf{uu}}$	$\mathcal{O}(1)$	$\mathcal{O}(n^2 M^2 d)$	$\mathcal{O}((n+d) M^2 \tilde{\ell}^2)$
$C_{\mathbf{fu}}$	$\mathcal{O}(\tilde{N} M \ell)$	$\mathcal{O}(n^2 \tilde{N} M \ell d)$	$\mathcal{O}((n+d)\tilde{N} M \tilde{\ell} \ell)$
$\text{diag}(k_{\mathbf{xx}})$	$\mathcal{O}(d \tilde{N} \ell^2)$	$\mathcal{O}((n+d)\tilde{N} \ell^2)$	$\mathcal{O}((n+d)\tilde{N} \ell^2)$
Lin. Alg.	$\mathcal{O}(\tilde{N} M^2)$	$\mathcal{O}(\tilde{N} M^2 + M^3)$	$\mathcal{O}(\tilde{N} M^2 + M^3)$

Table 1. Comparison of time complexities. M is the number of inducing variables, $\tilde{N}$ the batch size, d the number of channels in the time series, ℓ the length of the sequences, n the truncation level (for GPSig-IT and GPSig-IS) and $\tilde{l}$ the length of the inducing sequences. The last line of the table corresponds to linear algebra operations including matrix multiplication and matrix inversion.

6. Experiments

In this last section, we benchmark SigGPDE against GPSig-IT and GPSig-IS from Toth & Oberhauser (2020) on various multivariate time series classification tasks. For GPSig-IS,

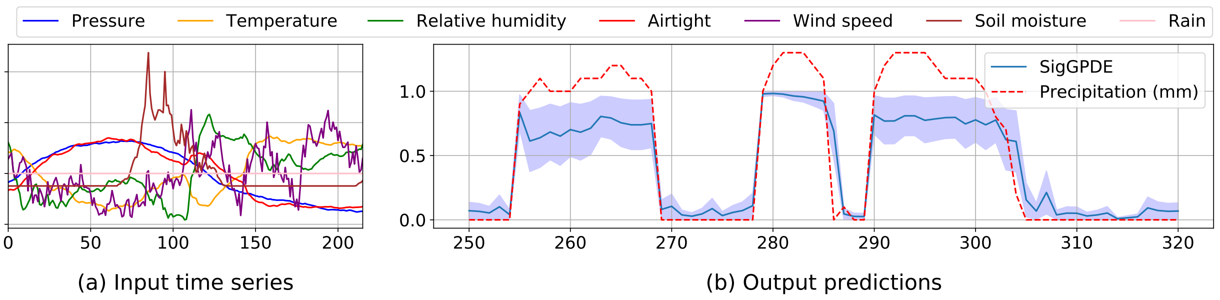

Figure 2. Weather forecast dataset. (a) One (standard scaled) multivariate time series $\mathbf{x}$ in input to the GP model. (b) Posterior mean of the SigGPDE GP when evaluated at multiple input time series like $\mathbf{x}$ on the test set. The actual precipitation amount is given for reference.

we use inducing sequences of length $\tilde{\ell} = 5$ as recommended by Toth & Oberhauser (2020). We highlight how SigGPDE performs competitively in terms of accuracy and uncertainty quantification but with a significant speed-up in the fitting compared to the other baselines. By comparing SigGPDE to GPSig-IT and GPSig-IS, we can benchmark different sparse approximation methods, whilst keeping the same GP model.

We use a mixture of UEA & UCR time series datasets (`timeseriesclassification.com`) and real world data for the final example. In the latter we discuss how the predictions provided by SigGPDE can be interpreted in a natural way via the interpretability of the interated integrals defining the signature and discussed in Sec. 3.1.

We measure the classification accuracy on the test set, assess the uncertainty quantification with mean negative log-predictive probabilities (NLPP) and report the runtime per-iteration. For each dataset all models are trained 3 times using a random training-validation split. The validation split is used to monitor the NLPP when optimizing the hyperparameters of the models. Further details on the training procedure can be found in the appendix. All code is written in TensorFlow using GPFlow (De G. Matthews et al., 2017).

6.1. Classifying digits in sequential MNIST

We start with a handwritten digit classification task, where writers were asked to draw the digits from 0 to 9. The instances are made up of 2-d trajectories of the pen traced across a digital screen. The trajectories are of length $\ell = 8$. The training and test sets are of size 7 494 and 3 498 respectively. We made use of $M = 500$ inducing features. In the results reported in Table 2, SigGPDE achieves even better accuracy and NLPP than the GPSig baselines, whilst being almost twice as fast than GPSig-IT.

6.2. Detecting whale call signals

In this example the task is to classify audio signals and distinguish one emitted from right whales from noise. The dataset (called RightWhaleCalls in the UEA archive) contains 10 934 train cases and 5 885 test cases. The signals are one-dimensional, sampled at 2kHz over 2 seconds, hence of length 4 000. We tackle this problem as a multivariate time series classification task, by taking the spectrogram of the univariate audio signal. The resulting streams are made of 29 channels corresponding to selected frequencies and are 30 time steps long. The results in Table 3 are obtained with $M = 700$ and show the significant speed-up of SigGPDE by almost one order of magnitude compared to GPSig. This speed-up is compensated by a minimal decrease in performance both in terms of accuracy and NLPP.

Table 2. Classification for sequential MNIST (PenDigits)

Model	Mean Acc.	NLPP	Time
GPSig-IS	97.42 ± 0.17	0.096 ± 0.005	0.186 (s/iter)
GPSig-IT	96.66 ± 0.59	0.115 ± 0.018	0.036 (s/iter)
SigGPDE	97.73 ± 0.13	0.085 ± 0.001	0.022 (s/iter)

Table 3. Classification for whale call signals

Model	Mean Acc.	NLPP	Time
GPSig-IS	86.97 ± 0.11	0.367 ± 0.005	0.438 (s/iter)
GPSig-IT	87.70 ± 0.42	0.357 ± 0.003	0.048 (s/iter)
SigGPDE	86.76 ± 0.36	0.382 ± 0.002	0.008 (s/iter)

6.3. Large scale classification of satellite time series

This is our large scale classification example on 1 million time series. The time series in this dataset represent a vegetation index, calculated from remote sensing spectral data. The 24 classes represent different land cover types (Petitjean et al., 2012). The aim in classifying these time series of length $\ell = 46$ is to map different vegetation profiles to

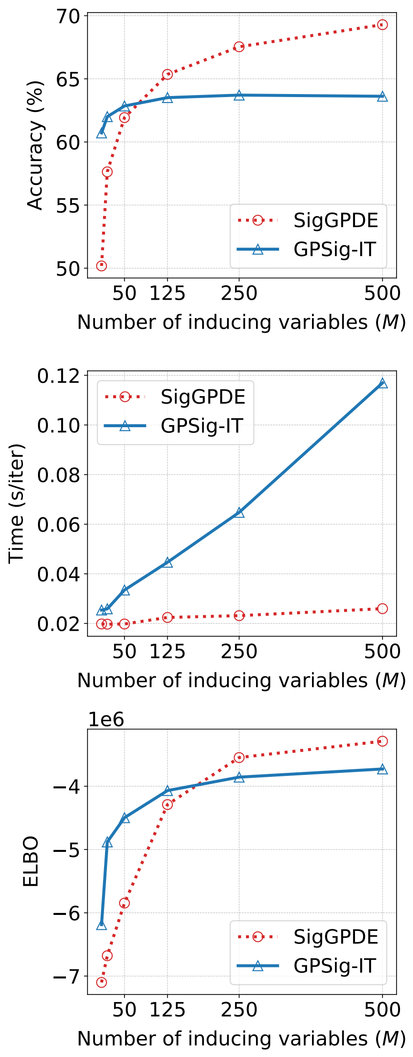

Figure 3. Large scale (1M) classification of satellite time series. Comparison of various metrics as functions of inducing variables.

different types of crops and forested areas. Due to the sheer size of this dataset we only compare SigGPDE to GPSig-IT as GPSig-IS is not scalable to such large dataset. In Fig. 3 we report the accuracy, time per iteration and ELBO by progressively increasing the number of inducing variables. In the regime with very few inducing variables, GPSig-IT outperforms SigGPDE. However, as the number of inducing features are increases, SigGPDE catches up and outperforms its competitor in all monitored metrics.

6.4. Weather forecast

In this last example we will be using a dataset of climatic variables recorded by the Max Planck Institute for Biogeochemistry[3] in the weather stations of WS Beutenberg and WS Saaleaue from 2004-2020. The data consists of 7-dimensional time series recorded once per 10 minutes where each channel represents a weather feature such as temperature, pressure, humidity etc. The goal is to predict whether it will rain over the next hour from the trajectory of all other features in the preceding 6 hours. To obtain binary labels for the classification task we set the label to 1 if the precipitation is larger than 1mm and to 0 otherwise. The inference mechanism is depicted on Fig. 2

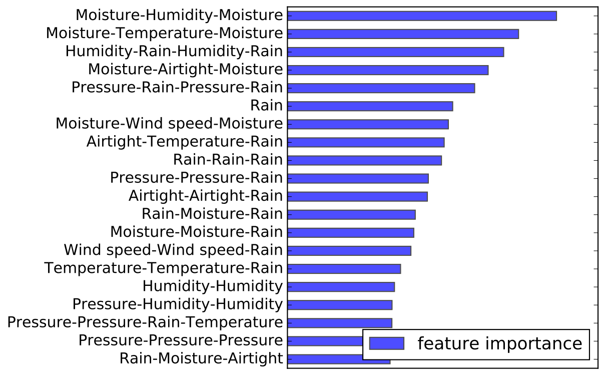

Figure 4. Top 20 signature features (by importance) used by SigGPDE to predict the probability of rainfull in the next hour from previous weather data. Each feature is a term in the signature. For example *Moisture-Humidity-Moisture* means that a change in the moisture channel followed by a change in the humidity channel and a change in the moisture channel is an important pattern.

A key feature proper to our model SigPDE is its interpretability. Looking at the variational mean vector $\mathbf{m}$ in eq. (29), we can extract the terms with highest relevance learned by the model. As discussed in Sec. 3.1, thanks to the corresponding signature features it is possible to infer which signature features used by the GP are more responsible for the produced outcome. The most relevant predictive features for this weather forecast experiment are represented in Fig. 4.

7. Conclusion

In this paper we have developed SigGPDE, a framework to perform variational inference for GP models on sequential data with orthogonal signature features. Firstly, we constructed inducing variables so that their covariance matrix is diagonal. Secondly, we showed that the gradients of the signature kernel are solutions of a hyperbolic PDE. As a result the ELBO is cheap to evaluate as gradient descent does not require backpropagating through the operations of the PDE solver. We benchmarked SigGPDE against the state-of-the-art GPSig on different time series classification tasks, showing a significant speed up and similar performance.

[3]https://www.bgc-jena.mpg.de/wetter/

References

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. Convergence of sparse variational inference in gaussian processes regression. *The Journal of Machine Learning Research*, 21:1–63, 2020a.

Burt, D. R., Rasmussen, C. E., and van der Wilk, M. Variational orthogonal features. *arXiv preprint arXiv:2006.13170*, 2020b.

Cass, T., Lyons, T., Salvi, C., and Yang, W. Computing the full signature kernel as the solution of a goursat problem. *arXiv preprint arXiv:2006.14794*, 2020.

Chevyrev, I. and Kormilitzin, A. A primer on the signature method in machine learning. *arXiv:1603.03788*, 2016.

De G. Matthews, A. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. Gpflow: A gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304, 2017.

Dozat, T. Incorporating nesterov momentum into adam. 2016.

Dutordoir, V., Durrande, N., and Hensman, J. Sparse gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*, pp. 2793–2802. PMLR, 2020.

Hensman, J., Durrande, N., and Solin, A. Variational fourier features for gaussian processes. *The Journal of Machine Learning Research*, 18(1):5537–5588, 2017.

Kanagawa, M., Hennig, P., Sejdinovic, D., and Sriperumbudur, B. K. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

Király, F. J. and Oberhauser, H. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20, 2019.

Lemercier, M., Salvi, C., Damoulas, T., Bonilla, E. V., and Lyons, T. Distribution regression for continuous-time processes via the expected signature. In *Artificial Intelligence and Statistics*, 2021.

Lyons, T. Rough paths, signatures and the modelling of functions on streams. *Proceedings of the International Congress of Mathematicians, Korea*, 2014.

Lyons, T. e. a. Coropa computational rough paths (software library). 2010. URL http://coropa.sourceforge.net/.

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

Lyons, T. J., Caruana, M., and Lévy, T. *Differential equations driven by rough paths*. Springer, 2007.

Matthews, A. G. d. G. *Scalable Gaussian process inference using variational methods*. PhD thesis, University of Cambridge, 2017.

Moore, P., Lyons, T., Gallacher, J., Initiative, A. D. N., et al. Using path signatures to predict a diagnosis of alzheimer's disease. *PloS one*, 14(9), 2019.

Petitjean, F., Inglada, J., and Gançarski, P. Satellite image time series analysis under time warping. *IEEE transactions on geoscience and remote sensing*, 50(8):3081–3095, 2012.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for machine learning*. The MIT Press, 2006.

Reizenstein, J. and Graham, B. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv preprint arXiv:1802.08252*, 2018.

Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., and Woods, E. Tslearn, a machine learning toolkit for time series data. *Journal of Machine Learning Research*, 21(118):1–6, 2020. URL http://jmlr.org/papers/v21/20-091.html.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Toth, C. and Oberhauser, H. Bayesian learning from sequential data using gaussian processes with signature covariances. In *International Conference on Machine Learning*, pp. 9548–9560. PMLR, 2020.

A. Additional Proof

In this section we prove Thm. 4.1 which yields an efficient algorithm to compute the gradients of the signature kernel with respect to its input paths. We recall Thm. 4.1 below.

Theorem 4.1. *For any $\gamma \in \mathcal{X}$ the directional derivative $k_\gamma(X,Y)$ of the signature kernel along the path γ satisfies the following relation*

$$k_\gamma(X,Y) = \int_0^T \int_0^T U(s,t)\widetilde{U}(T-s,T-t)(\dot{\gamma}_s^T \dot{Y}_t)dsdt$$

where $\widetilde{U}(s,t) = k(\overleftarrow{X}|_{[0,s]}, \overleftarrow{Y}|_{[0,t]})$ and where $\overleftarrow{X}, \overleftarrow{Y}$ are respectively the paths X, Y reversed in time.

Before proving Thm. 4.1 we need the following important lemma.

Lemma A.1. *For any two paths continuous paths of bounded variation $X, Y \in \mathcal{X}$ the signature kernel satisfies the following relation*

$$k(X,Y) = k(\overleftarrow{X}, \overleftarrow{Y}) \tag{43}$$

where $\overleftarrow{X}, \overleftarrow{Y}$ are the respectively X, Y reversed in time.

Proof. It is a standard result in rough path theory (see for example (Lyons et al., 2007)) that $S(\overleftarrow{X}) = S(X)^{-1}$, where the inverse is taken in the set of grouplike elements, which is a group. The operator on grouplike elements $g : S(X) \mapsto S(X)^{-1}$ reverses the order of the letters in each word and multiplies the result by -1 if the length of the word is odd. Expanding out $k(\overleftarrow{X}, \overleftarrow{Y})$ coordinate-wise it is easy to see that the two -1's for words of odd length cancel as multiplied together, therefore the expansion of $k(X,Y)$ matches the one of $k(\overleftarrow{X}, \overleftarrow{Y})$. □

Recall the notation for the signature kernel and its directional derivative used in the statement of Thm. 4.1:

$$U(s,t) := k\left(X|_{[0,s]}, Y|_{[0,t]}\right)$$
$$U_\gamma(s,t) := k_\gamma\left(X|_{[0,s]}, Y|_{[0,t]}\right)$$

Proof of Theorem 4.1. Let $\gamma : [0,T] \to \mathbb{R}^d$ be a continuous path of bounded variation along which we wish to differentiate k. Let's assume that for any $s,t \in [0,T]$ there exists a function $A_{s,t} : [0,T] \times [0,T] \to \mathbb{R}$ such that

$$U_\gamma(s,t) = \int_0^s \int_0^t A_{s,t}(u,v)U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv \tag{44}$$

Differentiating U_γ with respect to s and t we get

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = \int_0^s \int_0^t \frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv + A_{s,t}(s,t)U(s,t)(\dot{\gamma}_s^T \dot{Y}_t) \tag{45}$$

By eq. (41) we know that the directional derivative of the signature kernel along the path γ solves the following PDE

$$\frac{\partial^2 U_\gamma}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t)U_\gamma(s,t) + (\dot{\gamma}_s^T \dot{Y}_t)U(s,t) \tag{46}$$

Equating eq. (45) and eq. (46) we deduce that $A_{s,t}(s,t) = 1$ and

$$\begin{aligned}
&\int_0^s \int_0^t \frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv \\
&= U_\gamma(s,t)(\dot{X}_s^T \dot{Y}_t) \\
&= (\dot{X}_s^T \dot{Y}_t) \int_0^s \int_0^t A_{s,t}(u,v)U(u,v)(\dot{\gamma}_u^T \dot{Y}_v)dudv
\end{aligned}$$

Which implies that

$$\frac{\partial^2 A_{s,t}(u,v)}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t)A_{s,t}(u,v) \tag{47}$$

Or equivalently, by integrating with respect to s and t

$$A_{s,t}(u,v) = 1 + \int_u^s \int_v^t A_{s',t'}(u,v)(\dot{X}_{s'}^T \dot{Y}_{t'})ds'dt' \tag{48}$$

Hence

$$A_{T,T}(u,v) = \langle S(X)_{[u,T]}, S(Y)_{[v,T]} \rangle \tag{49}$$
$$= k(\overleftarrow{X}|_{[0,T-u]}, \overleftarrow{Y}|_{[0,T-v]}) \tag{50}$$

where the last equality is a consequence of Lemma A.1. Pluging back this result into eq. (44) concludes the proof. □

B. Additional Experimental Details

In this section we describe the experimental setup for Sec. 6. All experiments were conducted on NVIDIA Tesla P100 GPUs.

B.1. Data collection process

The classification tasks of Sections 6.1 and 6.2 were performed on two datasets (PenDigits, RightWhaleCalls) from the UCR & UEA time series classification repository.[4] For the large scale classification experiment of Sec. 6.3 we used a dataset of 1M satellite time series (STS).[5] Lastly, the climatic data (WeatherForecast) for rainfall prediction task in Sec. 6.4 was downloaded from the Max Planck Institute for Biogeochemistry website.[6]

[4]https://timeseriesclassification.com
[5]https://cloudstor.aarnet.edu.au/plus/index.php/s/pRLVtQyNhxDdCoM
[6]https://www.bgc-jena.mpg.de/wetter/weather_data.html

Data pre-processing included the following two steps. As explained in Sec. 3.1, we first add a monotonically increasing coordinate to all multivariate time series that we call "time", which effectively augments by one the number of channels. This is a standard procedure employed within signature based methods (Toth & Oberhauser, 2020; Chevyrev & Kormilitzin, 2016). Then, we standard scale the time series using tslearn library (Tavenard et al., 2020). This is particularly important for the WeatherForecast dataset where channels have different scales. Additional processing steps have been performed for two datasets (RightWhaleCalls, WeatherForecast) which we treat separately next.

A standard data transformation to tackle classification tasks on audio signals consists in computing their *spectrograms*. We follow this procedure for the RightWhaleCalls dataset which contains univariate highly-oscillatory time series of length $2\,000$. We used the scipy Python library to do so. The spectrogram is commonly represented as a graph with one axis representing time, the other axis representing the frequency, and the color intensity representing the amplitude of a particular frequency at a particular time. In this paper, we consider the spectrogram as a multivariate time series, where each channel represents the change in amplitude of a particular frequency over time. Furthermore, exploiting the fact that frequencies in whale call signals are typically between 50 and 300Hz, we only consider frequencies which fall within this range. As a result we obtain 28-dimensional time series each of length 30. We then apply the pre-processing steps described above.

To create the WeatherForecast dataset we used the recordings of various climatic variables in two weather stations located in Germany from 2004 to 2020. The outliers were filtered out, and we used the recordings of 7 variables (depicted on Fig. 2) over 6 hours in order to predict whether it would rain by more than 1mm over the next hour. There is one recording every 10min resulting in input time series of length $\ell = 36$. Since there were much fewer positive cases (raining) than negative cases (not raining), we dropped at random a fraction of the data, such that the ratio of positive/negative examples is brought down to 3.

B.2. Training procedure

The datasets for classification of sequential digits (PenDigits), audio signals (RightWhaleCalls), and satellite time series (STS) come with a predefined test-train split. In order to report standard deviations on our results we subsampled 20% (PenDigits,RightWhaleCalls) or 2% (STS) of the training set to form a validation set.

The training was equally split into 3 different phases. During the first phase, only the variational parameters are trained. For the second phase, both the variational parameters and the hyperparameters of the kernel are trained. During the last phase the variational parameters are trained on the full training set (the validation data being merged back). Overall, the hyperparameters are fixed for two-third of the iterations. SigGPDE and the GPSig-IT/IS baselines have the same set of hyperparameters, which correspond to the scaling factors for each channel for the ARD parametrization of the signature kernel Sec. 3.3. Those were initialized with the same value for all models. The inducing tensors for GPSig-IT and inducing sequences for GPSig-IS were initialized following the procedure outlined in (Toth & Oberhauser, 2020). We recall that for SigGPDE there is no such parameters to initialize. As recommended in (Toth & Oberhauser, 2020), we use a truncation level of $n = 4$ for their signature kernel algorithm (GPSig-IT/IS).

The minibatch size is either 50 (PenDigits, RightWhaleCalls) or 200 (STS). We used the Nadam optimizer (Dozat, 2016) with learning rate 10^{-3}. In Sec. 6 we report the time per iteration which corresponds to one minibatch.

C. Additional Algorithmic Details

In this section we start by outlining the space and time complexities of the algorithms underlying SigGPDE. Then, we explain how we have developed a dedicated CUDA TensorFlow operator for GPU acceleration to speed-up the computation of the signature kernel and its gradients.

C.1. Complexity analysis

The main algorithms underpinning SigGPDE consist in computing three different covariance matrices to evaluate the ELBO. These are the covariance matrix between the inducing variables $\mathbf{u}$ (denoted by $C_{\mathbf{uu}}$), between the marginal $\mathbf{f}$ and the inducing variables (denoted by $C_{\mathbf{fu}}$), and finally the covariance matrix of $\mathbf{f}$ (its diagonal is denoted by $\mathrm{diag}(k_{\mathbf{xx}})$). In Tables 4 and 5 we compare the time and space complexities for the corresponding SigGPDE algorithms to those of GPSig-IT/IS.

In the SigGPDE sparse variational inference framework, $C_{\mathbf{uu}}$ is diagonal which lowers both the memory and computational costs (see first line Tables 4 and 5). Besides there is no need to compute the Cholesky decomposition of $C_{\mathbf{uu}}$ to invert it (see last line Table 4). Lastly, in SigGPDE the inducing variables do not depend on any variational parameter (see last line Table 5).

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$C_{\mathbf{uu}}$	$\mathcal{O}(1)$	$\mathcal{O}(n^2M^2d)$	$\mathcal{O}((n+d)M^2\tilde{\ell}^2)$
$C_{\mathbf{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell d)$	$\mathcal{O}((n+d)\tilde{N}M\tilde{\ell}\ell)$
$\mathrm{diag}(k_{\mathbf{xx}})$	$\mathcal{O}(d\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$	$\mathcal{O}((n+d)\tilde{N}\ell^2)$
Lin. Alg.	$\mathcal{O}(\tilde{N}M^2)$	$\mathcal{O}(\tilde{N}M^2+M^3)$	$\mathcal{O}(\tilde{N}M^2+M^3)$

Table 4. Comparison of time complexities. M is the number of inducing variables, $\tilde{N}$ the batch size, d the number of channels in the time series, ℓ the length of the sequences, n the truncation level (for GPSig-IT and GPSig-IS) and $\tilde{\ell}$ the length of the inducing sequences.

Operation	SigGPDE (ours)	GPSig-IT	GPSig-IS
$C_{\mathbf{uu}}$	N/A	$\mathcal{O}(n^2M^2)$	$\mathcal{O}(M^2\tilde{\ell}^2)$
$C_{\mathbf{fu}}$	$\mathcal{O}(\tilde{N}M\ell)$	$\mathcal{O}(n^2\tilde{N}M\ell)$	$\mathcal{O}(\tilde{N}M\ell\tilde{\ell})$
$\mathrm{diag}(k_{\mathbf{xx}})$	$\mathcal{O}(\tilde{N}\ell^2)$	$\mathcal{O}(\tilde{N}\ell^2)$	$\mathcal{O}(\tilde{N}\ell^2)$
$\mathbf{z}$	N/A	$\mathcal{O}(n^2Md)$	$\mathcal{O}(M\tilde{\ell}d)$

Table 5. Comparison of space complexities, separated by algorithm to compute each covariance matrix. The last line accounts for the storage of the inducing tensors and inducing sequences in GPSig-IT and GPSig-IS.

C.2. Computing the signature kernel and its gradients

Recall that the signature kernel solves the following PDE,

$$\frac{\partial^2 U}{\partial s \partial t} = (\dot{X}_s^T \dot{Y}_t)U \qquad U(0,\cdot)=1,\ U(\cdot,0)=1 \tag{51}$$

therefore each kernel evaluation amounts to a call to a PDE solver. Using a straightforward implementation of a finite-difference PDE solver which consists in applying an update of the form

$$U(s_i,t_j) = g(U(s_{i-1},t_{j-1}), U(s_i,t_{j-1}), U(s_{i-1},t_j)),$$

in row or column order, the time complexity for N kernel evaluations for time series with d channels of length ℓ is $\mathcal{O}(dN\ell^2)$. Indeed there is no data dependencies between each of the N kernel evaluations, hence we can solve each PDE in parallel. But, this does not reduce the quadratic complexity with respect to the length ℓ. However, it is possible to parallelize the PDE solver by observing that instead of solving the PDE in row or column order, we can update the antidiagonals of the solution grid. As illustrated on Fig. 5, each cell on an antidiagonal can be updated with in parallel as there is no data dependency between them. Therefore, we propose a CUDA implementation where N collections of $2\ell-1$ threads (the number of cells on the biggest antidiagonal) running in parallel can simultaneously update an antidiagonal of the solution grids.

To compute the gradients, we use the result from Thm. 4.1. During the forward pass we solve the PDEs defined by the input time series using the CUDA operator described

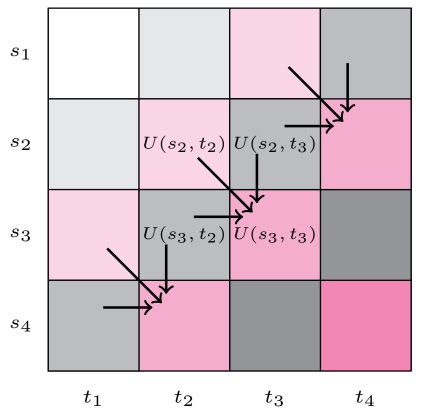

Figure 5. Parallelization of the finite-difference scheme. Each cell on an antidiagonal can be computed in parallel, provided the previous antidiagonals have been computed.

above. For the backward pass, we first solve the PDEs with the input time series reversed in time, by calling the same CUDA operator. Second, we compute the gradients using simple vectorized TensorFlow operations.