Bayesian Learning from Sequential Data using Gaussian Processes with Signature Covariances

Csaba Toth [1] **Harald Oberhauser** [1]

Abstract

We develop a Bayesian approach to learning from sequential data by using Gaussian processes (GPs) with so-called signature kernels as covariance functions. This allows to make sequences of different length comparable and to rely on strong theoretical results from stochastic analysis. Signatures capture sequential structure with tensors that can scale unfavourably in sequence length and state space dimension. To deal with this, we introduce a sparse variational approach with inducing tensors. We then combine the resulting GP with LSTMs and GRUs to build larger models that leverage the strengths of each of these approaches and benchmark the resulting GPs on multivariate time series (TS) classification datasets.

1. Introduction

The evolution of some state variable, parameter or object gives naturally gives rise to *sequential data*, which is defined by having a notion of order on the incoming information. The ordering relation, or *index set* does not have to represent physical time, but for simplicity we will call it as such. For example, besides time series, sources of sequential data are text (Pennington et al., 2014), DNA (Heather & Chain, 2016), or even topological data analysis (Chevyrev et al., 2018). This ubiquity of sequential data has received special attention by the machine learning community in recent years. This paper is motivated by the following three approaches:

Deep learning approaches. Deep learning approaches, such as the celebrated LSTM network (Hochreiter & Schmidhuber, 1997), other forms of RNNs (Cho et al., 2014) and convolutional networks have successfully been applied to a variety of tasks involving sequential data (Sutskever et al., 2014; Oord et al., 2016). Deep learning models can approximate any continuous function, but the cost is a large number of parameters, high variance and poor interpretability. This leaves the door open for alternative approaches not only as competitors, but as complementary building blocks in a larger model.

Bayesian approaches. Often not only point predictions, but estimates of the associated uncertainties are required (Ghahramani, 2013). GPs (Rasmussen & Williams, 2006) provide flexible priors over functions of the data in nonparametric Bayesian models. In the context of sequential data, two prominent ways to use GPs are: (1) using as covariance functions kernels specifically designed for sequences (Lodhi et al., 2002; Cuturi, 2011; Cuturi & Doucet, 2011; Al-Shedivat et al., 2017), (2) modelling the evolution in a latent space, that emits the observations, as a discrete dynamical system with a GP prior on the transition function, a model called the Gaussian Process State Space Model (GPSSM) (Frigola et al., 2013; 2014; Mattos et al., 2016; Eleftheriadis et al., 2017; Doerr et al., 2018; Ialongo et al., 2019). These two approaches are not mutually exclusive; if one models the latent system as a higher order Markov process, then sequence kernels can incorporate the effect of past states.

Signature approaches. The *signature feature map* is a well-developed tool from stochastic analysis that represents a sequence as an element in a linear space of tensors, (Chen, 1958; Lyons et al., 2007). While not a mainstream machine learning approach, it is gaining attention since it can represent *non-linear* functions of sequences as *linear* functions of signature features, and can be made invariant to parametrization similar to dynamic time warping (DTW). For example, (Kidger et al., 2019) use them as layer in a deep learning architecture; (Kiraly & Oberhauser, 2019) introduce kernels for sequences by taking inner products of signature features; (Chevyrev & Oberhauser, 2018) use them for maximum mean discrepancies between laws of stochastic processes. In particular, if $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel for vector-valued data, then (Kiraly & Oberhauser,

[1]Mathematical Institute, University of Oxford, Oxford, United Kingdom. Correspondence to: Csaba Toth <csaba.toth@maths.ox.ac.uk>, Harald Oberhauser <harald.oberhauser@maths.ox.ac.uk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

2019) uses signatures to derive the following kernel

$$\mathrm{k}(\mathbf{x},\mathbf{y}) = \sum_{m=0}^{M} \sigma_m^2 \sum_{\mathbf{i}_m,\mathbf{j}_m} c(\mathbf{i}_m)c(\mathbf{j}_m) \prod_{l=1}^{m} \Delta_{i_l,j_l}\kappa(\mathbf{x}_{i_l},\mathbf{y}_{j_l}),$$

for two sequences $\mathbf{x} = (\mathbf{x}_i)_{i=1}^{l_\mathbf{x}}$ and $\mathbf{y} = (\mathbf{y}_j)_{j=1}^{l_\mathbf{y}}$, with the double difference operator defined as $\Delta_{i,j}\kappa(\mathbf{x}_i,\mathbf{y}_j) := \kappa(\mathbf{x}_{i+1},\mathbf{y}_{j+1}) - \kappa(\mathbf{x}_i,\mathbf{y}_{j+1}) - \kappa(\mathbf{x}_{i+1},\mathbf{y}_j) + \kappa(\mathbf{x}_i,\mathbf{y}_j)$, and the sums are taken over multi-indices $\mathbf{i}_m = (i_1,\dots,i_m)$ with $1 \le i_1 \le \cdots \le i_m < l_\mathbf{x}$ (and analogous for $\mathbf{j}_m$) and some explicitly computable coefficients $c(\mathbf{i}) \in [0,1]$.

Our contribution. In principle, one can just use the signature kernel and algorithms from (Kiraly & Oberhauser, 2019) as covariance to define a GP for sequential data. However, the computational complexity becomes quickly prohibitive and the low-rank approximation too crude, which ultimately does not lead to competitive results on many TS benchmarks. We therefore develop a different approach to signature covariances that builds on two recent advances in GP inference, namely variational inference (Titsias, 2009; Hensman et al., 2015; Matthews et al., 2016) and inter-domain inducing points (Lázaro-Gredilla & Figueiras-Vidal, 2009) to alleviate the computational burden. In particular, we show that one can use sparse tensors as inter-domain inducing points by optimizing a variational bound. Moreover, we use this GP as a building block in combination with RNNs to build models that combine the strenghts of these different tools. This results in scalable inference algorithms and we use this to benchmark on standard TS datasets (i) against popular non-Bayesian time series classifiers purely in terms of accuracy, (ii) against alternative Bayesian models by comparing the calibration of uncertainties for predictions. Code and benchmarks are publically available at `http://github.com/tgcsaba/GPSig`.

2. Background and Notation

Given data $(\mathbf{X}, Y)$ consisting of $n_\mathbf{X}$ inputs $\mathbf{X} = (\mathbf{x}_1,\dots,\mathbf{x}_{n_\mathbf{X}}) \subset \mathcal{X}$ with labels $Y = (y_1,\dots,y_{n_\mathbf{X}}) \in \mathbb{R}$, a common Bayesian approach is to put a prior on a set of functions $\{f | f : \mathcal{X} \to \mathbb{R}\}$, update this prior by conditioning on $(\mathbf{X}, Y)$, and then use the resulting posterior to make inference about the label $y_\star$ of an unseen point $\mathbf{x}_\star$. When this is done with Gaussian priors, the central object is a GP $f = (f_\mathbf{x})_{\mathbf{x}\in\mathcal{X}}$ which is specified by mean and covariance function. Throughout we interchangibly use the notation $f_\mathbf{x}$ and $f(\mathbf{x})$. Below we recall how covariances can be constructed from feature maps and discuss the case when $\mathcal{X}$ is a space of sequences of arbitrary length.

The feature space view. Given a map $\varphi : \mathcal{X} \hookrightarrow V$ that injects $\mathcal{X}$ into a linear space V, a natural way to put a prior on a function class $\mathcal{X} \to \mathbb{R}$ is to consider *linear* functions of φ as model, that is $f(\mathbf{x}) := \langle \ell, \varphi(\mathbf{x}) \rangle$ to model $f(\mathbf{x}_i) \approx y_i$ for some "weights" $\ell \in V$. Uncertainty about f is then specified by uncertainty about ℓ (and the hyperparameters of φ). We refer to φ as a *feature map* and to V as *feature space*. Throughout we assume that $f = (f_\mathbf{x})_{\mathbf{x}\in\mathcal{X}}$ is a centered GP and predictions about unseen points can then be made by Gaussian conditioning. If the task is classification where the labels Y are discrete, such an approach can be still applied by using a GP $f = (f_\mathbf{x})_{\mathbf{x}\in\mathcal{X}}$ as *nuiscance function* to put a *prior on the class membership probability* by specifying $p(y = 1|\mathbf{x}) = \sigma(f(\mathbf{x}))$ where σ is for example a sigmoid.

Polynomial features. The classical example is $\mathcal{X} = \mathbb{R}^d$ and

$$\varphi(\mathbf{x}) := (1, \mathbf{x}, \mathbf{x}^{\otimes 2}, \mathbf{x}^{\otimes 3}, \dots, \mathbf{x}^{\otimes M}) \tag{1}$$

where $\mathbf{x}^{\otimes m} \in (\mathbb{R}^d)^{\otimes m}$ is a tensor. We recall background on tensors in Appendix A. If we set $f(\mathbf{x}) = \langle \ell, \varphi(\mathbf{x}) \rangle$ and put a centered Gaussian prior on $\ell = (\ell_1,\dots,\ell_M)$, then by linearity of the tensor product and expectation it follows that

$$\mathbb{E}[f_\mathbf{x} f_\mathbf{y}] = 1 + \textstyle\sum_{m=1}^{M} \langle \Sigma_m^2, \mathbf{x}^{\otimes m} \otimes \mathbf{y}^{\otimes m} \rangle$$

where $\Sigma_m^2 := \mathbb{E}[\ell_m \otimes \ell_m] \in (\mathbb{R}^d)^{\otimes(2m)}$. Taking Σ_m^2 to be an isotropic diagonal matrix $\sigma_m^2 \cdot I^{\otimes m}$ recovers the polynomial kernel,

$$\mathbb{E}[f_\mathbf{x} f_\mathbf{y}] = \textstyle\sum_{m=0}^{M} \sigma_m^2 \langle \mathbf{x}, \mathbf{y} \rangle^m$$

where we use the convention $\langle \mathbf{x}, \mathbf{y} \rangle^0 = \sigma_0^0 = 1$. Many variations exist, for example other classes of polynomials, such as Hermite polynomials (the eigenfunctions of the classic RBF kernel), can increase the effectiveness, since they allow to make the associated feature expansion infinite dimensional. However, what makes any such class of polynomials a sensible choice for φ is that by the Stone–Weierstrass theorem, any continuous compactly supported function $\mathcal{X} \to \mathbb{R}$ can be arbitrary well approximated as linear functions of $\varphi(\mathbf{x})$. This approximation property often runs under the name *universality* (Micchelli et al., 2006; Sriperumbudur et al., 2011).

Sequences as paths. We study the case when one observation $\mathbf{x}$ is a sequence of $l_\mathbf{x}$ tuples, $\mathbf{x} = (x_i, t_i)_{i=1,\dots,l_\mathbf{x}}$, and each tuple (x_i, t_i) specifies that at "time" t_i a vector $x_i \in \mathbb{R}^d$ was measured. We denote with $\mathcal{X}_{seq}$ the set of all such sequences. and emphasize that the length $l_\mathbf{x} \ge 1$ is not fixed, which is a common case in real-world data. We now introduce an even larger set than $\mathcal{X}_{seq}$: sequential data evolves in discrete time but often arises by sampling a quantity that evolves in continuous time. Thus above the set $\mathcal{X}_{seq}$ of sequences lurks the larger set of finite horizon paths

$$\mathcal{X}_{paths} = \{\mathbf{x} \in C([0, t_\mathbf{x}], \mathbb{R}^d) : t_\mathbf{x} \in \mathbb{R}^+, \mathbf{x}(0) = 0, \|\mathbf{x}\|_{bv} < \infty\},$$

which are simply continuous $\mathbb{R}^d$-valued functions on some bounded time-interval. Here $\|\mathbf{x}\|_{bv} := \sup_{0\le t_1<\cdots<t_n\le t_{\mathbf{x}}} \sum_{i=1}^n |\mathbf{x}(t_{i+1}) - \mathbf{x}(t_i)|$ denotes the usual bounded variation norm[1]. The set $\mathcal{X}_{seq}$ naturally embeds into $\mathcal{X}_{paths}$ by mapping a sequence $\mathbf{x} \in \mathcal{X}_{seq}$ to the path that is given by linear interpolation between the points $(0,0), (t_1, x_1), \ldots, (t_{l_{\mathbf{x}}}, x_{l_{\mathbf{x}}})$; formally $\mathbf{x} \in \mathcal{X}_{seq}$ maps to the element of $\mathcal{X}_{paths}$ defined as

$$t \mapsto (t_{i+1} - t_i)^{-1} \left(x_i(t_{i+1} - t) + x_{i+1}(t - t_i)\right) \quad (2)$$

for $t \in [t_i, t_{i+1})$. Henceforth, we implicitly use this embedding, i.e. with slight abuse of notation, given $\mathbf{x} \in \mathcal{X}_{seq}$ we also write $\mathbf{x} \in \mathcal{X}_{paths}$. Key to our approach, and what makes it different to classic approaches, is to define a GP indexed by the larger set $\mathcal{X}_{paths}$ rather than just $\mathcal{X}_{seq}$. At first, this looks wasteful since $\mathcal{X}_{paths}$ is much bigger than $\mathcal{X}_{seq}$ and in practice one only has access to discrete time data (already for storage reasons). But on a theoretical side, going from discrete time to continuous time has two big advantages: (i) by construction, such a GP is consistent in the high-frequency limit (that is when we sample an object evolving in continuous time at higher and higher frequencies), (ii) we can make use of well-developed theory from stochastic analysis; in particular we use so-called signature features for paths.

3. From signature features to covariances

The signature feature map Φ can be seen as a generalization of the polynomial feature map φ as defined in (1) from the domain $\mathcal{X} = \mathbb{R}^d$ of vectors to the domain of paths $\mathcal{X}_{paths}$. It is defined as

$$\Phi(\mathbf{x}) = (1, \textstyle\int_0^{t_{\mathbf{x}}} d\mathbf{x}, \int_0^{t_{\mathbf{x}}} d\mathbf{x}^{\otimes 2}, \ldots, \int_0^{t_{\mathbf{x}}} d\mathbf{x}^{\otimes m}) \quad (3)$$

where $\int_0^t d\mathbf{x}^{\otimes(m+1)} := \int_0^{t_{\mathbf{x}}} \int_0^s d\mathbf{x}^{\otimes m} \otimes d\mathbf{x}(s) \in (\mathbb{R}^d)^{\otimes m}$ and $\int_0^t d\mathbf{x}^{\otimes 1} := \mathbf{x}(t)$. Signatures are classic objects in stochastic analysis, but probably unfamiliar to researchers in ML and we provide background in Appendix B. Additionally, we recommend (Chevyrev & Kormilitzin, 2016) for a hands-on introduction to signature features that provides a good complement to our presentation, motivating signatures as a generalization of polynomial features to sequences.

For what follows, only three facts will be used about $\mathbf{x} \mapsto \Phi(\mathbf{x})$: (1) it maps paths of different length to the same space, thus makes paths of different length comparable, (2) functions of paths can be arbitrary well approximated by *linear* functions of $\Phi(\mathbf{x})$, (3) it distinguishes paths that follow different trajectories, but not paths that only differ by parametrization). These points explain why signature features $\Phi(\mathbf{x})$ are a natural generalization of polynomial features $\varphi(\mathbf{x})$: not only do they use the same feature space (sequences of tensors), they also have the same attractive properties such as being able to approximate continuous functions. Below we discuss how to make $\mathbf{x} \mapsto \Phi(\mathbf{x})$ distinguish paths with different time-parametrizations.

[1]Our approach generalizes to much rougher paths, such as Brownian trajectories and we give details in Appendix B.

	Vectors	Paths
Domain	$\mathbf{x} \in \mathbb{R}^d$	$\mathbf{x} = (\mathbf{x}_t)_{t\in[0,t_{\mathbf{x}}]} \in \mathcal{X}_{paths}$
Features	$\varphi(\mathbf{x}) = (\mathbf{x}^{\otimes m})_m$	$\Phi(\mathbf{x}) = (\int d\mathbf{x}^{\otimes m})_m$
Feature space	$\prod_m (\mathbb{R}^d)^{\otimes m}$	$\prod_m (\mathbb{R}^d)^{\otimes m}$
Functions	$f : \mathbb{R}^d \to \mathbb{R}$	$f : \mathcal{X}_{paths} \to \mathbb{R}$
Covariance	$\sum_m \sigma_m^2 \langle \mathbf{x}, \mathbf{y}\rangle^m$	$\sum_m \sigma_m^2 \int\int \langle d\mathbf{x}_s, d\mathbf{y}_t\rangle$

Table 1: Comparison of polynomial and signature features

Parametrization (in)variance. A classic empirical finding that led to DTW is that functions of sequences are to a certain degree invariant to the time parametrization: for example, different speakers pronounce words at different speeds. However, sometimes the parametrization matters, e.g. for financial data. Thus we do not only care about the set of functions

$$\{f : \mathcal{X}_{paths} \to \mathbb{R} \,|\, \text{cont. and compactly supported}\} \quad (4)$$

but also about the subset of it that consist of parametrization invariant functions. To make this precise, we call $\mathbf{x} = (x_t)$ a *reparametrization* of $\mathbf{y} = (y_t)$ if there exists a a smooth increasing function $\rho : [0, t_{\mathbf{x}}] \to [0, t_{\mathbf{y}}]$ (the "time change") such that $x_t = y_{\rho(t)}$ for all t. We call an element f of (4) *parametrization invariant* if $f(\mathbf{x}) = f(\mathbf{y})$ when $\mathbf{x}$ and $\mathbf{y}$ are reparametrizations. Often the function we want to learn is invariant to a bit of reparametrization but not extreme reparametrization, so we need a more nuanced way to quantify parametrization (in)variance. Hence, what we really want is a hyperparameter $\tau \ge 0$ that signifies the degree of parametrization invariance: for $\tau = 0$ all the mass of our prior should concentrate on the "extreme case" that is the subset of (4) consisting of functions that are parametrization invariant; and as τ gets increased the probability mass should spread out to functions that are sensitive to parametrization. This would allow to infer the degree of parametrization invariance by automatic relevance discovery (ARD). To accomplish this, we parametrize signature features with a parameter $\tau \ge 0$ as follows

$$\Phi_\tau(\mathbf{x}) := (1, \textstyle\int_0^{t_{\mathbf{x}}} d\mathbf{x}_\tau, \int_0^{t_{\mathbf{x}}} d\mathbf{x}_\tau^{\otimes 2}, \ldots, \int_0^{t_{\mathbf{x}}} d\mathbf{x}_\tau^{\otimes m})$$

where $\mathbf{x}_\tau(t) := (\tau \cdot t, \mathbf{x}(t)) \in \mathbb{R}^{1+d}$. This simply makes the parametrization part of the trajectory $\{\mathbf{x}_\tau(t) : t \in [0, t_{\mathbf{x}}]\} \subset \mathbb{R}^d$ by adding an extra coordinate. Since signatures distinguish different trajectories (but not the speed at which we run through them), it follows that for $\tau > 0$,

$$\Phi_\tau(\mathbf{x}) = \Phi_\tau(\mathbf{y}) \text{ if and only if } \mathbf{x} = \mathbf{y},$$

and for $\tau = 0$ we have,

$$\Phi_0(\mathbf{x}) = \Phi_0(\mathbf{y}) \text{ if and only if } \mathbf{x} \sim \mathbf{y}$$

since the extra coordinate is "switched off". Here, $\sim$ denotes tree-like equivalence, but we invite the reader to read $\mathbf{x} \sim \mathbf{y}$ as saying that $\mathbf{x}$ is a reparametrization of $\mathbf{x}$. This is strictly speaking not true and we give the precise mathematical statement in Appendix B, but note that for real-world data, tree-like equivalence is synonymous with reparametrization.

Signature covariances. Following the feature space view, we now argue in complete analogy to the case of the classical polynomial feature map φ for $\mathcal{X} = \mathbb{R}^d$, and define a centered GP $f = (f_{\mathbf{x}})_{\mathbf{x} \in \mathcal{X}_{paths}}$ by putting a centered Gaussian prior on $\ell = (\ell_1, \dots, \ell_M)$ and setting

$$f_{\mathbf{x}} := \langle \ell, \Phi_\tau(\mathbf{x}) \rangle.$$

The GP is hence fully specified by its covariance function

$$k(\mathbf{x}, \mathbf{y}) = \sum_{m=0}^{M} \langle \Sigma_m^2, \int d\mathbf{x}_\tau^{\otimes m} \otimes \int d\mathbf{y}_\tau^{\otimes m} \rangle \qquad (5)$$

that has $(\tau, M, \Sigma_1^2, \dots, \Sigma_M^2)$ as hyperparameters. In particular, choosing an isotropic covariance structure for Σ_m^2 gives

$$k(\mathbf{x}, \mathbf{y}) = \sum_{m=0}^{M} \int \int \sigma_m^2 \langle d\mathbf{x}_\tau, d\mathbf{y}_\tau \rangle^m,$$

where the integrations are over the simplices $0 < t_1 < \cdots < t_m < t_{\mathbf{x}}$ and $0 < s_1 < \cdots < s_m < t_{\mathbf{y}}$. Furthermore, for the special case of paths that arise as linear interpolation of sequences $\mathbf{x}, \mathbf{y} \in \mathcal{X}_{seq}$, this covariance reduces to iterated sums of increments

$$k(\mathbf{x}, \mathbf{y}) = \sum_{m=0}^{M} \sum_{\mathbf{i}_m, \mathbf{j}_m} \sigma_m^2 c(\mathbf{i}_m, \mathbf{j}_m) \prod_{l=1}^{m} \langle \Delta \mathbf{x}_{i_l}, \Delta \mathbf{y}_{j_l} \rangle \qquad (6)$$

for some explicitly known constants $c(\mathbf{i}_m, \mathbf{j}_m)$, where the inner sums are over all m-tuples $\mathbf{i}_m = (i_1, \dots i_m)$ and $\mathbf{j}_m = (j_1, \dots, j_m)$ with $i_1 \leq \cdots \leq i_m$ and $j_1 \leq \cdots \leq j_m$, and the convention that the empty summation is equal to 1.

GP regularity. One expects that the GP with covariance (5) has nice regularity properties, however, the index set $\mathcal{X}_{paths}$ is a very large space so some care is needed. In Appendix C, we compute covering numbers that yield explicit bounds on the modulus of continuity in terms of the path path length $\|\mathbf{x}\|_{bv}$.

Theorem 1. *Let $L > 0$ and $\mathcal{X}_{paths}^L := \{\mathbf{x} \in \mathcal{X}_{paths} : \|\mathbf{x}\|_{bv} \leq L\}$. There exists a centered GP $f = (f_{\mathbf{x}})_{\mathbf{x} \in \mathcal{X}_{paths}^L}$ with $k(\mathbf{x}, \mathbf{y})$ as defined in (5) as covariance function and that has continuous sample paths $\mathbf{x} \mapsto f_{\mathbf{x}}$. Further, an explicit bound on its modulus of continuity in terms of L is given in equation (14).*

We now have a well-defined GP for Bayesian inference for sequences at hand that inherits many of the attractive properties of signature features. To turn this into useful models for large TS benchmarks we develop efficient inference algorithms in the next section.

4. Sparse variational inducing tensors

To reiterate, we are given data $(\mathbf{X}, Y)$ consisting of $n_{\mathbf{X}}$ sequences $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_{n_{\mathbf{X}}})$ of maximal length $l_{\mathbf{X}} := \max_{\mathbf{x} \in \mathbf{X}} l_{\mathbf{x}} \subset \mathcal{X}_{seq}$ that evolve in $\mathbb{R}^d$ with labels $Y = (y_1, \dots, y_{n_{\mathbf{X}}})$, and the task is to predict labels $y_\star$ of unseen points $\mathbf{x}_\star$. For sequential data, the sample size $n_{\mathbf{X}}$ and associated covariance matrix inversion is not the only compational bottleneck but also the maximal length of sequences $l_{\mathbf{X}}$, and the dimension d of the state space matter: $n_{\mathbf{X}}$, $l_{\mathbf{X}}$ and d can be simultaneously large.

In this section, we introduce a sparse inference scheme to approximate the posterior of our GP, that locates the inducing points in a space other than the data-domain; an approach that is usually coined the term *inter-domain* sparse variational inference (Lázaro-Gredilla & Figueiras-Vidal, 2009; Matthews et al., 2016). This allows for more efficient data-representation and faster inference. Key to our approach is that signature features take values in a well-understood subset of the feature space $\prod_{m \geq 0} (\mathbb{R}^d)^{\otimes m}$. This allows as us to augment the index set with structured tensors, and locate inducing points in this larger index set.

Variational inference. As is well-known, inference for GPs scales as $O(n_{\mathbf{X}}^3)$, see Section 3.3. in (Rasmussen & Williams, 2006). This first led to *sparse* models, (Quiñonero-Candela & Rasmussen, 2005), that select a subset $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_{n_{\mathbf{Z}}}\}$ of $\mathbf{X}$ consisting of $n_{\mathbf{Z}} \ll n_{\mathbf{X}}$ points, and subsequently to *pseudo-inputs*, (Snelson & Ghahramani, 2006), that select points $\mathbf{Z}$ that are not necessarily in $\mathbf{X}$. This was a big step towards complexity reduction, but pseudo-inputs are prone to overfitting, (Matthews, 2017). A different idea is to treat $\mathbf{Z}$ as parameters of a variational approximation (Titsias, 2009) and not as model parameters; that is the points $\mathbf{Z}$ are choosen simultaneously with the hyperparameters of the GP by maximising a lower bound on the log-marginal likelihood $\log p(Y)$, the so-called *evidence lower bound* (ELBO), given as

$$\log p(Y) \geq \mathbb{E}_{q(f_{\mathbf{X}})}[\log p(Y|f_{\mathbf{X}})] - D_{KL}\left[q(f_{\mathbf{Z}}) \parallel p(f_{\mathbf{Z}})\right], \qquad (7)$$

where $f_{\mathbf{X}}$ and $f_{\mathbf{Z}}$ denotes the GP evaluated at the data-points and the inducing locations. Typically, $q(f_{\mathbf{Z}})$ is given a free-form multivariate Gaussian to be learnt from the data, and then extended to other indices of the GP by *prior conditional matching*, i.e. $q(f_{\mathbf{X}}|f_{\mathbf{Z}}) = p(f_{\mathbf{X}}|f_{\mathbf{Z}})$. Initially applied to regression, this was extended to classification (Chai, 2012;

Hensman et al., 2015). Among its advantages are that it gives a nonparametric approximation to the true posterior, adding inducing points only improves the approximation, and any optimization method can be used to maximize the ELBO, most importantly, stochastic optimization; see (Hensman et al., 2013; Bauer et al., 2016; Bui et al., 2017).

Inter-domain approaches. Another idea is to go beyond the original index set and place inducing points $\mathbf{Z}$ in a different space $\mathcal{X}'$, that is, given a centered GP $g = (g_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}}$ one augments the original index set $\mathcal{X}$ by a set $\mathcal{X}'$ to define a new GP $(g_{\mathbf{x}})_{\mathbf{x}\in\mathcal{X}\cup\mathcal{X}'}$ and then locates the inducing points in this bigger model. This was suggested in (Lázaro-Gredilla & Figueiras-Vidal, 2009) in the context of integral transforms, which was extended in (Hensman et al., 2016), and studied in more generality in (Matthews et al., 2016). In general, it is not obvious how to find a useful augmentation set $\mathcal{X}'$ and define the covariance enlarged to $\mathcal{X}\cup\mathcal{X}'$.

4.1. A sparse variational tensor augmentation.

Given any GP with a covariance function $\mathrm{k}(\mathbf{x},\mathbf{y}) := \langle\Phi(\mathbf{x}),\Phi(\mathbf{y})\rangle$ where Φ is explicitly known[2], we propose that a natural augmentation candidate is the "feature space" $\mathcal{X}' := \mathrm{span}\{\Phi(\mathbf{x}) : \mathbf{x}\in\mathcal{X}\}$ itself. The covariance function k of g can be simply extended to $\mathcal{X}\cup\mathcal{X}'$ by linearity,

$$\mathrm{k}(\mathbf{x},\mathbf{z}) := \mathrm{k}(\mathbf{z},\mathbf{x}) := \alpha\,\mathrm{k}(\mathbf{x},\mathbf{x}') + \beta\,\mathrm{k}(\mathbf{x},\mathbf{x}'') \qquad (8)$$

for $\mathbf{x}\in\mathcal{X}$, $\mathbf{z} = \alpha\Phi(\mathbf{x}') + \beta\Phi(\mathbf{x}'') \in \mathcal{X}'$, $\alpha,\beta\in\mathbb{R}$; analogous for $\mathrm{k}(\mathbf{z},\mathbf{z}')$ with $\mathbf{z},\mathbf{z}'\in\mathcal{X}'$. For our GP,

$$\mathcal{X}' = \mathrm{span}\{\Phi_\tau(\mathbf{x}) : \mathbf{x}\in\mathcal{X}_{paths}\} \subset \textstyle\prod_{m\geq 0} V^{\otimes m}$$

where we denote $V := \mathbb{R}^d$. We can thus extend our signature covariance (5) to $\mathcal{X}_{seq}\cup\prod_{m\geq 0}V^{\otimes m}$ by (8). This provides a flexible class of inducing point locations $\mathbf{Z}$ by optimizing over elements of the tensor algebra $\mathbf{Z}\subset\prod_{m\geq 0}V^{\otimes m}$. We coin these inducing point locations as *inducing tensors*.

Consistency of augmentation. A subtle point about augmenting the index set is that maximizing the ELBO in (7) is not necessarily equivalent anymore to minimizing a rigorously defined KL divergence between the true posterior process and its approximation over the unaugmented index set. In (Matthews et al., 2016), a sufficient condition given for this to hold is that the prior GP evaluated at the newly added indices is deterministic conditioned on the original GP. In the case of (8), this is easily seen to be true, since the augmented indices arise as linear combinations of elements in the original index set. Therefore, the corresponding GP evaluations arise as linear combinations of evaluations of the original process by the fact that the feature space $\mathcal{X}'$ is a *representation* of the *Hilbert space generated by the process* (Berlinet & Thomas-Agnan, 2003).

Representation of inducing tensors. We define our sparse inducing tensors as

$$\mathbf{z} = (z_m)_{m=0,\dots,M} \in \textstyle\prod_{m=0}^M V^{\otimes m},$$

where $z_0\in\mathbb{R}$ and $z_m = v_{m,1}\otimes v_{m,2}\otimes\cdots\otimes v_{m,m}$ for $m\geq 1$. We remark that this construction does not generally give tensors that can be signatures of paths. However, they can be represented as linear combinations of signatures, hence the previous argument about the augmentation carries over. Also, informally, what gives the data-efficiency of inducing tensors is exactly that they are not represented in a basis of signatures, but as sparse tensors.

By linearity of integration and the inner product, the inducing point covariance $\mathbb{E}[f_{\mathbf{z}}f_{\mathbf{z}'}]$ equals the inner product

$$\textstyle\sum_{m=0}^M \sigma_m^2\langle z_m, z'_m\rangle = \sum_{m=0}^M \sigma_m^2\prod_{k=1}^m\langle v_{m,k}, v'_{m,k}\rangle \qquad (9)$$

the cross-covariance $\mathbb{E}[f_{\mathbf{x}}f_{\mathbf{z}}] = \sum_{m=0}^M\sigma_m^2\langle\Phi_m(\mathbf{x}), z_m\rangle$,

$$\langle\Phi_m(\mathbf{x}), z_m\rangle = \textstyle\int\langle dx_{t_1}, v_{m,1}\rangle\cdots\langle dx_{t_m}, v_{m,m}\rangle$$

where the integration is over the simplex $0 < t_1 < \cdots < t_{l_{\mathbf{x}}}$. Finally, note that we just need to evaluate the above for piecewise linear paths since these are the only paths that arise via the embedding (2), $\mathcal{X}_{seq}\hookrightarrow\mathcal{X}_{paths}$. For such paths, the above integrals reduce to iterated sums, hence $\langle\Phi_m(\mathbf{x}), z_m\rangle$ equals

$$\sum_{\mathbf{i}} c(\mathbf{i})\langle x_{i_1+1} - x_{i_1}, v_{m,1}\rangle\cdots\langle x_{i_m+1} - x_{i_m}, v_{m,m}\rangle \qquad (10)$$

where the sum is taken over all m-tuples $\mathbf{i} = (i_1,\dots,i_m)$ of the form $1\leq i_1\leq\cdots\leq i_m\leq l_{\mathbf{x}}$ and $c(\mathbf{i})\leq 1$ is given by an explicit calculation. Similarly to (6), replacing $c(i_1,\dots,i_m)$ with 1 if there are no repeating indices in $(i_1,\dots,i_m)$ and otherwise with 0 gives a good approximation[3] . Below we use this approximation to (10) since it makes the recursive algorithms simpler but note that a simple modification exactly computes (10) for a marginal computational overhead.

4.2. Algorithms.

We need to compute the three covariance matrices: (1) $K_{\mathbf{ZZ}}$ of inducing tensors $\mathbf{Z}$ and inducing tensors $\mathbf{Z}$, (2) $K_{\mathbf{ZX}}$ of inducing tensors $\mathbf{Z}$ and sequences $\mathbf{X}$, (3) $K_{\mathbf{XX}}$ of sequences $\mathbf{X}$ and sequences $\mathbf{X}$. Using the above tensor representations allows to give vectorized algorithms for (1)

[2] Mercer's Theorem guarantees the existence of Φ, but not in a sufficiently explicit form.

[3] It converges to $\langle\Phi(\mathbf{x}),\mathbf{z}\rangle$ when the grid gets finer, $|t_{i+1}-t_i|\downarrow 0$, see (Chevyrev & Oberhauser, 2018) to (10).

and (2) in Algorithms 1 and 2, respectively. For (3) we use a modification of Algorithm 3 from (Kiraly & Oberhauser, 2019) which we recall in Appendix D.2. We use notation defined in Appendix D.1, which can be briefly summarized as: Σ denotes the slice-wise sum operator, $\boxplus$ the (forward) cumulative sum operator, $+1$ the shift operator, and $\odot$ denotes the element-wise product of arrays. Additionally, we set $\Delta x_i := x_{i+1} - x_i$ for $i \in \{1, \dots, l_{\mathbf{x}} - 1\}$. For $v, v' \in V$, d denotes the time to compute $\langle v, v' \rangle$, c the memory requirement of v.

Proposition 1. *Algorithm 1 computes the covariance matrix $K_{\mathbf{ZZ}}$ of $n_{\mathbf{Z}}$ inducing points in $O(M^2 \cdot n_{\mathbf{Z}}^2 \cdot d)$ steps. Algorithm 2 computes the cross-covariance matrix $K_{\mathbf{ZX}}$ in $O(M^2 \cdot n_{\mathbf{X}} \cdot n_{\mathbf{Z}} \cdot l_{\mathbf{X}} \cdot d)$ steps. Additionally to storing the inducing tensors $\mathbf{Z}$, Algorithm 1 requires $O(M^2 \cdot n_{\mathbf{Z}}^2)$ memory, Algorithm 2 requires $O(M^2 \cdot n_{\mathbf{X}} \cdot n_{\mathbf{Z}} \cdot l_{\mathbf{X}})$ memory.*

Proposition 1 follows by inspection of the algorithms and we emphasize the following points: (i) Both algorithms are linear in the maximal sequence length $l_{\mathbf{X}}$. (ii) M is a hyperparameter, and in all our experiments we learnt from the data $M \leq 5$, thus the quadratic complexity in M is negligible. (iii) The memory cost of inducing tensors $\mathbf{Z}$ is much less than for the data $\mathbf{X}$, which is stored in $O(n_{\mathbf{X}} \cdot l_{\mathbf{X}} \cdot d)$ memory, which is important because the inducing tensors are variational parameters, and not amenable to subsampling, while the learning inputs can be subsampled as noted by (Hensman et al., 2013). Especially for GPUs memory cost is decisive and such savings are very important.

The computation of $K_{\mathbf{XX}}$ detailed in Appendix D.2 has time complexity $O((M + d) \cdot n_{\mathbf{X}}^2 \cdot l_{\mathbf{X}}^2)$ and memory of $O(d \cdot n_{\mathbf{X}} \cdot l_{\mathbf{X}} + n_{\mathbf{X}}^2 \cdot l_{\mathbf{X}}^2)$. However, given a factorizing likelihood, one only requires $K_{\mathbf{X}} := [k(\mathbf{x}, \mathbf{x})]_{\mathbf{x} \in \mathbf{X}}$, which eliminates the quadratic cost in $n_{\mathbf{X}}$. It turns out that this is enough to train on GPUs with reasonable minibatch sizes (e.g. $n_{\mathbf{X}} = 50$) on several real world datasets. We remark that the low-rank algorithms in (Kiraly & Oberhauser, 2019) allow to trade off accuracy for linear cost in $l_{\mathbf{X}}$, but we found that using the full-rank algorithm performs much better, and the above will allow us to apply it to several datasets with great results. Finally, note that the ELBO (7) requires an additional matrix inversion and multiplication in $O(n_{\mathbf{Z}}^2 \cdot n_{\mathbf{X}} + n_{\mathbf{Z}}^3)$ time, which is not significant in our case.

Variations. The following variations produce a more flexible covariance function: (i) given a nonlinear function $\varphi : \mathbb{R}^d \hookrightarrow V$ into a linear space V, lift a sequence $\mathbf{x} = (t_i, x_i)$ to a path by taking the linear interpolation of $(0, 0), (t_1, \varphi(x_1)), \dots, (t_{l_{\mathbf{x}}}, \varphi(x_{l_{\mathbf{x}}}))$; with φ the identity on $\mathbb{R}^d$ this recovers the original embedding (2), (ii) adding lags is a classic time series pre-processing technique, justified by Takens' theorem, (Takens, 1981), that guarantees that attractors in a high-dimensional dynamical system can be reconstructed from low-dimensional observations. Both

Algorithm 1 Computing the inducing covariances $K_{\mathbf{ZZ}}$

Input: Tensors $\mathbf{Z} = (\mathbf{z}_i)_{i=1,\dots,n_{\mathbf{Z}}} \subset \prod_{n=0}^{m} V^{\otimes n}$, scalars $(\sigma_0^2, \sigma_1^2, \dots, \sigma_m^2)$, depth $m \in \mathbb{N}$
Compute $K[i, j, n, k] \leftarrow \langle v_{n,k}^i, v_{n,k}^j \rangle$ for $i, j \in \{1, \dots, n_{\mathbf{Z}}\}$, $n \in \{1, \dots, m\}$ and $k \in \{1, \dots, n\}$
Initialize $R[i, j] \leftarrow \sigma_0^2$ for $i, j \in \{1, \dots, n_{\mathbf{Z}}\}$
for $n = 1$ **to** m **do**
  Assign $A \leftarrow K[:, :, n, 1]$
  for $k = 2$ **to** n **do**
    Iterate $A \leftarrow K[:, :, n, k] \odot A$
  end for
  Update $R \leftarrow R + \sigma_n^2 \cdot A$
end for
Output: Matrix of inducing covariances $K_{\mathbf{ZZ}} \leftarrow R$

Algorithm 2 Computing the cross-covariances $K_{\mathbf{ZX}}$

Input: Tensors $\mathbf{Z} = (\mathbf{z}_i)_{i=1,\dots,n_{\mathbf{Z}}} \subset \prod_{n=0}^{m} V^{\otimes n}$, sequences $\mathbf{X} = (\mathbf{x}_i)_{i=1,\dots,n_{\mathbf{X}}} \subset \mathcal{X}_{seq}$, scalars $(\sigma_0^2, \sigma_1^2, \dots, \sigma_m^2)$, depth $M \in \mathbb{N}$
Compute $K[i, j, l, m, k] \leftarrow \langle v_{m,k}^i, \Delta x_{j,t_l} \rangle$ for $i \in \{1, \dots, n_{\mathbf{Z}}\}$, $j \in \{1, \dots, n_{\mathbf{X}}\}$, $l \in \{1, \dots, l_{\mathbf{X}} - 1\}$, $m \in \{1, \dots, M\}$ and $k \in \{1, \dots, m\}$
Initialize $R[i, j] \leftarrow \sigma_0^2$ for $i \in \{1, \dots, n_{\mathbf{Z}}\}$, $j \in \{1, \dots, n_{\mathbf{X}}\}$
for $m = 1$ **to** M **do**
  Assign $A \leftarrow K[:, :, n, 1]$
  for $k = 2$ **to** m **do**
    Iterate $A \leftarrow K[:, :, :, m, k] \odot A[:, :, \boxplus + 1]$
  end for
  Update $R \leftarrow R + \sigma_m^2 \cdot A[:, :, \Sigma]$
end for
Output: Matrix of cross-covariances $K_{\mathbf{ZX}} \leftarrow R$

points add non-linearities to the feature space which can make the learning more efficient. If the original sequence evolves in $\mathbb{R}^d$, this preprocessing results in a sequence (and then path) that evolves in a general high-dimensional space V. However, formulas (8), (9), and (10) show that only inner product evaluations on V are used and these can be computationally cheap even if V is high or even infinite dimensional. For example, following (Kiraly & Oberhauser, 2019) we may take a kernel $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ and use $\varphi(x) := \kappa(x, \cdot)$, to build a sequence $\kappa_{\mathbf{x}} \in V_{seq}$ in the RKHS of κ. The only change in complexity is to replace d in the big O-bounds by the cost of the kernel evaluation. Such extensions also increase the number of hyperparameters which can have adversarial effects, but in our experiments both extensions led generically to better results.

Table 2: Average ranks of GPs on datasets (Baydogan, 2015) with the 1st and 2nd best in bold and italicized for each row

	GP-SIG-LSTM	GP-SIG-GRU	GP-SIG	GP-LSTM	GP-GRU	GP-KCONV1D
MEAN RANK (NLPP, $n_{\mathbf{X}} < 300$)	*2.80*	2.90	**2.20**	4.70	4.00	4.40
MEAN RANK (ACC., $n_{\mathbf{X}} < 300$)	*3.00*	3.10	**2.80**	4.25	4.25	3.60
MEAN RANK (NLPP, $n_{\mathbf{X}} \geq 300$)	**2.33**	3.33	*2.83*	4.83	4.33	3.33
MEAN RANK (ACC., $n_{\mathbf{X}} \geq 300$)	**2.17**	3.50	*3.00*	4.17	4.33	3.83
MEAN RANK (NLPP, ALL)	*2.63*	3.06	**2.44**	4.75	4.13	4.00
MEAN RANK (ACC., ALL)	**2.69**	3.25	*2.88*	4.22	4.28	3.69

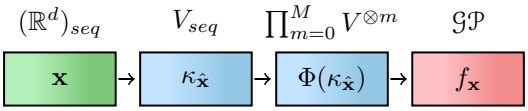

Figure 1: The GP-Sig model

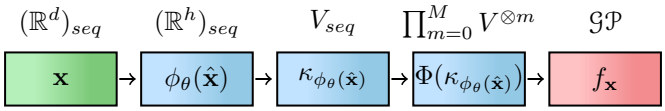

Figure 2: The GP-Sig-RNN model

5. Experiments

TS classification. Using GPFlow (de G. Matthews et al., 2017), Keras (Chollet et al., 2015), we implemented three models: GP-Sig, GP-Sig-LSTM, and GP-Sig-GRU. All three use the signature covariance with the sparse inducing tensors of Section 4. GP-Sig is a plain vanilla variational GP classifier. Previous applications of neural nets to covariance constructions, in particular (Wilson et al., 2016; Al-Shedivat et al., 2017), inspired GP-Sig-LSTM and GP-Sig-GRU that include an RNN as a sequence-to-sequence transformation with h hidden units; see Figures 1 and 2 where $\hat{\mathbf{x}}$ denotes augmentation with lags and $\kappa_{\hat{\mathbf{x}}}$ a static kernel as in above variation paragraph. We benchmarked these GP models on 16 multivariate TS classification datasets, a collection introduced in (Baydogan, 2015) that has become a semi-standard archive in TS classification, e.g. we cite 7 papers in Appendix E.4 that use these datasets. The same datasets are also used in (Ismail Fawaz et al., 2019) to compare several deep learning architectures for TSC.

As Bayesian baselines we used three GP models: (i) GP-LSTM and (ii) GP-GRU consist of an LSTM and a GRU network with an RBF kernel on top, in which case the RNNs are used as a sequence-to-vector transformation from $\mathbb{R}^d_{seq}$ to $\mathbb{R}^h$; (iii) GP-KConv1D uses the convolutional kernel introduced in (van der Wilk et al., 2017) in 1-dimension (time). Throughout we used sparse variational inference: for GP-Sig-LSTM, GP-Sig-GRU, GP-Sig, the inducing tensors detailed in Section 4 are used; for GP-LSTM and GP-GRU the inducing points are located in the output space of the RNN layer, $\mathbb{R}^h$; for GP-KConv1D, the inducing patches of (van der Wilk et al., 2017) are used.

We used $n_{\mathbf{Z}} = 500$ for all models[4]; further all use a static kernel in one form or another, which we fixed to be the RBF kernel. The signature kernel was truncated[5] at $M = 4$, and for GP-Sig $p = 1$ lags were used; the GP-Sig-RNNs did not use lags, as the sequence of hidden states already incorporate lagged information about past observations. The window size in GP-KConv-1D was set to $w = 10$. The RNN-architectures were selected independently for all models by grid-search among 6 variants, that is, the number of hidden units from $[8, 32, 128]$ and with or without dropout. For training, early stopping was used with $n = 500$ epochs patience; a learning rate of $\alpha = 1 \times 10^{-3}$; a minibatch size of 50; as optimizer Adam (Kingma & Ba, 2014) and Nadam (Dozat, 2015) were employed. Implementations are detailed in Appendix E.1, the datasets in Appendix E.2, the training and grid-search methodology in Appendix E.3.

Discussion of results. For GPs, we report accuracies and negative log-predictive probabilities (nlpp), the latter take not only accuracies, but the calibration of probabilities into account as well. Table 2 shows the average ranks among the GPs. The full table of nlpps and accuracies with mean and standard deviation over 5 model trains are reported in Appendix E.4 in Table 5 and Table 6. As non-Bayesian baselines, we report accuracies of eight frequentist TS classifiers in Table 7. On Figure 5, we visualize the box-plot distributions of (i) negative log-predictive probabilities of GPs, (ii) accuracies of both GPs and frequentist methods.

The signature models perform consistently the best in terms of average rankings of both nlpp and accuracy among the GPs. Particularly, they achieve stronger mean performance and a smaller variance across datasets. To explain this, inspecting the results in Tables 5, 6, we observe that all other GP baselines perform very poorly on some datasets, while the signature based models perform at least moderately well

[4]Using $n_{\mathbf{Z}} = 500$ is clearly superfluous for small datasets, which is fixed for the sake of consistent settings across datasets.

[5]For these experiments, the $M = 4$ value seemed to give an optimal trade-off between computational complexity and expressiveness of the kernel, see Appendix E.1 for more details.

on *all datasets*. We believe this ties in to the universality property of signatures, see Appendix B.5. The convolutional GP, GP-KConv1D, which also has a very small parameter set, performed rather competitively with the deep kernel baselines, even on larger datasets. Comparison among variants of GP-Sig can be summarized as follows: for smaller datasets ($n_{\mathbf{X}} < 300$), GP-Sig outperforms other variants as it has a very small parameter set; for larger datasets ($n_{\mathbf{X}} \geq 300$), GP-Sig-LSTM performs best which conforms with the intuition that RNNs suffer from small sample sizes. A related observation is that GP-LSTM and GP-GRU perform about on par, while GP-Sig-LSTM does much better than GP-Sig-GRU, which suggests that the signature makes explicit use of the additional gate in the LSTM network.

Compared only in terms of accuracy, GP-Sig competes with frequentist classifiers: it outperforms the usual DTW baseline and competes with state-of-the art classifiers such as MUSE and MLSTMFCN. Purely based on accuracy, these win overall, but the difference is usually small, hence the extra Bayesian advantages come at a small cost. Furthermore, since the MLSTMFCN is also a deep learning baseline, it would be interesting to see how it performs incorporated into a deep kernel, possibly used as a sequence-to-sequence transformation with the signature kernel on top. Obviously TS classification is a vast field and many other models could be considered; e.g. we did not use recurrent GPs or GPSSMs since (1) they have so-far not been used for TS classification, possibly because there is no sequential nature in the output space, (2) we did not find a GPflow implementation that would allow to use sequence kernels in the GP transition function. (Implementation of (Ialongo et al., 2019) does currently not allow taking subsequences of past states. An implementation would require much further work, but an interesting project would be to combine our models.)

Inducing tensors vs inducing sequences. Our results rely on the inter-domain approach using tensors to locate inducing points from Section 4. An alternative is to use sequences for the inducing points, $\mathbf{Z} \subset \mathcal{X}_{seq}$, and controlling their maximal length $l_{\mathbf{Z}} := \max_{\mathbf{z} \in \mathbf{Z}} l_{\mathbf{z}}$ to be of order m, i.e. $l_{\mathbf{Z}} \sim m$. We coin this approach *inducing sequences*. Intuitively, one expects the inducing tensors to be more efficient than inducing sequences, since they make full use of the structure of the signature feature space/covariance. To test this intuition, we compared the performance of the inducing tensors and inducing sequences subject to both having the same computational complexitiy. For this experiment, we took the AUSLAN dataset (Dua & Graff, 2017), which consists of $n_c = 95$ classes for $n_{\mathbf{X}} = 1140$ training examples. This is a challenging dataset as the inducing variables need to characterize the abundance of classification boundaries.

We used GP-Sig with the same settings as in the previous experiments. The hyperparameters of the kernel were a-priori learnt with $n_{\mathbf{Z}} = 500$ inducing tensors, and we purely investigated how the quality of the approximation changes for both approaches by varying the number of inducing points $n_{\mathbf{Z}}$. For each number of inducing variables, both approaches were trained independently 5 times for 300 epochs with random initialization of the inducing variables, for details on which see Appendix E.3. We plot on Figure 3 three metrics: (1) the achieved ELBO on the training set, (2) the achieved accuracy, and (3) nlpp on the testing set. At $n_{\mathbf{Z}} = 500$ both approaches are close to saturation, but the inducing tensors consistently perform better. We remark that in practice, an important aspect is also how well the kernel hyperparameters can be recovered, that we did not consider here, and is a tricky question for sparse variational inference in general (Bauer et al., 2016). Although, intuition suggests that the closer the model to saturation is with respect to the inducing points, the more consistent should the optimization be with un-sparsified variational inference.

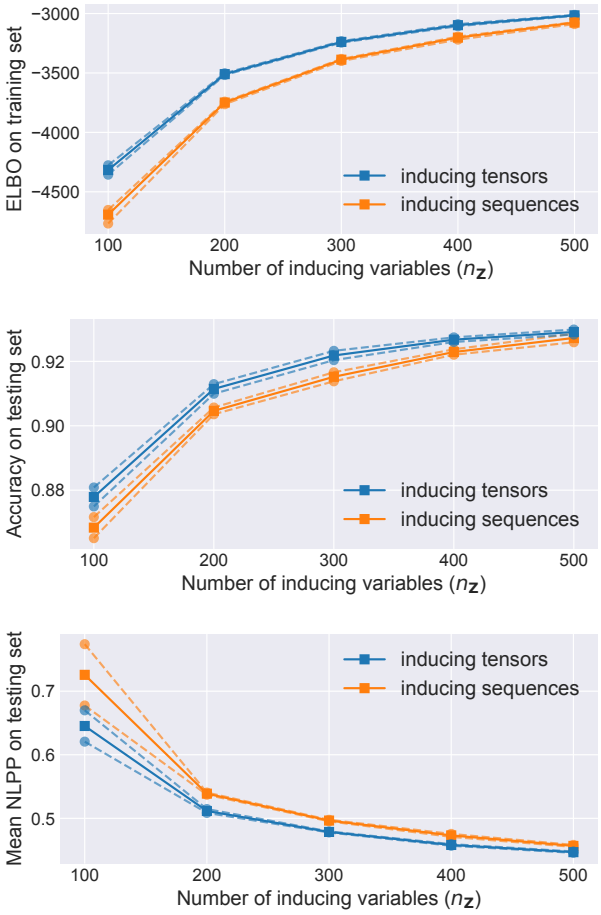

Figure 3: Achieved ELBO (top), accuracy (middle), mean nlpp (bottom) after 300 epochs of training the variational parameters with random initialization and pre-learnt kernel hyperparameters, that were treated as fixed. Solid is the mean over 5 independent runs, dashed is the 1-std region.

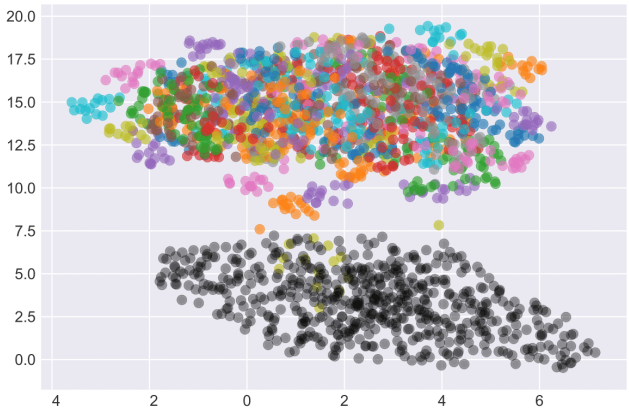

Figure 4: A UMAP visualization of the allocation of feature representations of data-points (coloured), and inducing tensors (black) in the feature space on the AUSLAN dataset.

Visualizing inducing tensors. To gain more intuition, we visualized the feature space for one of the trained models on AUSLAN with $n_{\mathbf{Z}} = 500$ inducing tensors. We used UMAP (McInnes et al., 2018) with the semi-metric $(\mathbf{x}, \mathbf{y}) \mapsto (\mathrm{k}(\mathbf{x}, \mathbf{x}) + \mathrm{k}(\mathbf{y}, \mathbf{y}) - 2\,\mathrm{k}(\mathbf{x}, \mathbf{y}))^{0.5}$ for $\mathbf{x}, \mathbf{y} \in \mathcal{X} \cup \mathcal{X}'$, see Figure 4. There are two imminent observations: (i) in the point cloud corresponding to the data, the classes hardly look linearly separable; (ii) the tensors, however, seem to live on a completely separate subspace than the data. The algorithm achieves 92% accuracy on this set, therefore, point (i) is likely due to information being lost in the projection. However, point (ii) challenges the intuition about classical sparse variational inference, that the inducing points are located mixed-in with the data-points, concentrating close to the classification boundaries (Hensman et al., 2015). In general, the mechanism of how inter-domain inducing points represent the information in the data seems to be more complicated than classically.

To explain point (ii), we remark that this phenomenon is not surprising at all: signature features live in a manifold that is embedded in the linear tensor space $\prod_{m=0}^{M} V^{\otimes m}$. In general, sparse tensors of the form (4.1) will *not* be signatures of paths. We believe variational inference works because of an interplay of two factors: Firstly, signatures of finite sequences can be written as finite linear combinations of such sparse tensors. Secondly, the prior conditional term used to define $q(f_{\mathbf{x}}|f_{\mathbf{Z}}) = p(f_{\mathbf{x}}|f_{\mathbf{Z}})$. The feature space is *congruent* to the prior GP (Berlinet & Thomas-Agnan, 2003), which means that for $\mathbf{x} \in \mathcal{X}_{seq}$, the value of $f_{\mathbf{x}}$ given $f_{\mathbf{Z}}$ is not only almost deterministic when $\mathbf{x}$ is close to any of $\mathbf{z} \in \mathbf{Z}$, but when it is close to any linear combinations of elements in $\mathbf{Z}$. By the first remark this can always be achieved given a large enough $n_{\mathbf{Z}}$. To sum up, the inducing tensors do not represent signature features individually, but form atomic building blocks such that their linear combinations induce the actual variational posterior at the data-examples.

6. Conclusion

We used a classical object from stochastic analysis – signatures – to define a GP for sequential data. The GP inherits many of the theoretical guarantees that are known for signature features such as universality and parametrization invariance. To make it scalable, we develop "inducing tensors" that exploit the structure of the feature space, inter-domain inducing points, and variational inference. Applied in a plain vanilla variational framework, this yields a classifier, GP-Sig, that is not only competitive in terms of nlpp with other GP models, but also with state-of-the-art frequentist TS classifiers in terms of accuracy alone. As one of our reviewers remarked, several datasets we consider have a strong signal-to-noise ratio, which makes it worthwhile to point out that even for such datasets, the alternative GP baselines suffer on at least some of them, while the proposed models are consistently able to learn on all datasets. This observation ties in to the universality property, and it suggests that GPs with signatures can be a good starting point when building Bayesian models on time series datasets.

We also demonstrate that signatures can be used as a building block in deep kernels to build larger GP models that leverage the benefits of both, RNNs and signatures. Interestingly, we find that the vanilla GP-Sig model outperforms the GP-Sig-RNNs for smaller datasets, conforming to the intuition that smaller sample sizes are detrimental for recurrent neural nets. To really get the best of both worlds, one could insert an additional model selection step, that specifies whether a parametric transformation is layer used before feeding the input into the kernel or not. Alternatively, it could also be possible to increase the flexibility of the sequential GP model while staying within a purely nonparametric framework using deep GPs (Damianou & Lawrence, 2013) by e.g. applying a GP layer as observation-wise state-space embedding before the kernel computation. The inference framework of (Salimbeni et al., 2019) for deep GPs could also come in handy when moving to datasets with lower signal-to-noise ratios, which can require GP models capable of handling not only epistemic (reducible) uncertainty, but aleatoric (irreducible) uncertainty in the data. It would also be interesting to see if such sequence kernels can be used to improve recurrent GP models (Mattos et al., 2016; Ialongo et al., 2019) by incorporating sequential information into the GP transition function, that could potentially allow for a more efficient latent state representation.

Acknowledgments

CT is supported by the "Mathematical Institute Award" by the University of Oxford, HO is supported by the EPSRC grant "Datasig" [EP/S026347/1], the Alan Turing Institute, and the Oxford-Man Institute. CT and HO would like to thank the reviewers for helpful and constructive comments.

References

Adler, R. and Taylor, J. *Random Fields and Geometry*. Springer Monographs in Mathematics. Springer New York, 2009. ISBN 9780387481166. URL https://books.google.co.uk/books?id=R5BGvQ3ejloC.

Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. Learning scalable deep kernels with recurrent structure. *J. Mach. Learn. Res.*, 18(1):2850–2886, January 2017. ISSN 1532-4435.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse Gaussian process approximations. In *Advances in neural information processing systems*, pp. 1533–1541, 2016.

Baydogan, M. *Multivariate Time Series Classification Datasets*, 2015. URL http://mustafabaydogan.com. [Accessed: 2020-02-05].

Baydogan, M. G. and Runger, G. Learning a symbolic representation for multivariate time series classification. *Data Mining and Knowledge Discovery*, 29(2):400–422, 2015a.

Baydogan, M. G. and Runger, G. C. Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, 30:476–509, 2015b.

Berlinet, A. and Thomas-Agnan, C. *Reproducing Kernel Hilbert Spaces in Probability and Statistics*. Springer US, 2003. ISBN 9781402076794. URL https://books.google.co.uk/books?id=v79sBNG34coC.

Bui, T. D., Yan, J., and Turner, R. E. A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation. *Journal of Machine Learning Research*, 18(104):1–72, 2017. URL http://jmlr.org/papers/v18/16-603.html.

Chai, K. M. A. Variational multinomial logit Gaussian process. *Journal of Machine Learning Research*, 13(Jun): 1745–1808, 2012.

Chen, K.-T. Integration of paths—a faithful representation of paths by non-commutative formal power series. *Trans. Amer. Math. Soc.*, 89:395–407, 1958.

Chevyrev, I. and Kormilitzin, A. A Primer on the Signature Method in Machine Learning. *ArXiv e-prints*, March 2016.

Chevyrev, I. and Oberhauser, H. Signature moments to characterize laws of stochastic processes. *arXiv preprint 1810.10971*, 2018. URL https://arxiv.org/abs/1810.10971.

Chevyrev, I., Nanda, V., and Oberhauser, H. Persistence Paths and Signature Features in Topological Data Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018. doi: 10.1109/tpami.2018.2885516.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

Chollet, F. et al. Keras. https://github.com/fchollet/keras, 2015.

Cuturi, M. Fast global alignment kernels. pp. 929–936, 01 2011.

Cuturi, M. and Doucet, A. Autoregressive Kernels For Time Series. *arXiv e-prints*, art. arXiv:1101.0673, Jan 2011.

Damianou, A. and Lawrence, N. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pp. 207–215, 2013.

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. *Journal of Machine Learning Research*, 18:40:1–40:6, 2017.

Doerr, A., Daniel, C., Schiegg, M., Duy, N.-T., Schaal, S., Toussaint, M., and Sebastian, T. Probabilistic recurrent state-space models. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1280–1289, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Dozat, T. Incorporating Nesterov Momentum into Adam. 2015.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Dudley, R. M. Sample functions of the gaussian process. In *Selected Works of RM Dudley*, pp. 187–224. Springer, 2010.

Eleftheriadis, S., Nicholson, T., Deisenroth, M., and Hensman, J. Identification of gaussian process state space models. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5309–5319. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7115-identification-of-gaussian-process-state-space-models.pdf.

Frigola, R., Lindsten, F., Schön, T. B., and Rasmussen, C. E. Bayesian inference and learning in gaussian process state-space models with particle mcmc. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 3156–3164. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5085-bayesian-inference-and-learning-in-gaussian-process-state-space-models-with-particle-mcmc.pdf.

Frigola, R., Chen, Y., and Rasmussen, C. E. Variational gaussian process state-space models. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 3680–3688, Cambridge, MA, USA, 2014. MIT Press.

Ghahramani, Z. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 371:20110553, 02 2013. doi: 10.1098/rsta.2011.0553.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Hambly, B. and Lyons, T. Uniqueness for the Signature of a path of bounded variation and the reduced path group. *Ann. of Math. (2)*, 171(1):109–167, 2010. ISSN 0003-486X. doi: 10.4007/annals.2010.171.109. URL http://dx.doi.org/10.4007/annals.2010.171.109.

Heather, J. M. and Chain, B. The sequence of sequencers: The history of sequencing dna. *Genomics*, 107(1):1 – 8, 2016. ISSN 0888-7543. doi: https://doi.org/10.1016/j.ygeno.2015.11.003. URL http://www.sciencedirect.com/science/article/pii/S0888754315300410.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. *CoRR*, abs/1309.6835, 2013.

Hensman, J., de G. Matthews, A. G., and Ghahramani, Z. Scalable variational gaussian process classification. In Lebanon, G. and Vishwanathan, S. V. N. (eds.), *AISTATS*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015. URL http://dblp.uni-trier.de/db/conf/aistats/aistats2015.html#HensmanMG15.

Hensman, J., Durrande, N., and Solin, A. Variational fourier features for gaussian processes, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Ialongo, A. D., Van Der Wilk, M., Hensman, J., and Rasmussen, C. E. Overcoming mean-field approximations in recurrent Gaussian process models. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2931–2940, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/ialongo19a.html.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.*, 33(4):917–963, July 2019. ISSN 1384-5810. doi: 10.1007/s10618-019-00619-1. URL https://doi.org/10.1007/s10618-019-00619-1.

Karim, F., Majumdar, S., Darabi, H., and Harford, S. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237 – 245, 2019. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.04.014. URL http://www.sciencedirect.com/science/article/pii/S0893608019301200.

Karlsson, I., Papapetrou, P., and Boström, H. Generalized random shapelet forests. *Data Min. Knowl. Discov.*, 30(5):1053–1085, September 2016. ISSN 1384-5810. doi: 10.1007/s10618-016-0473-y. URL https://doi.org/10.1007/s10618-016-0473-y.

Khoshnevisan, D. *Multiparameter processes: an introduction to random fields*. Springer Science & Business Media, 2002.

Kidger, P., Bonnier, P., Perez Arribas, I., Salvi, C., and Lyons, T. Deep signature transforms. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 3099–3109. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8574-deep-signature-transforms.pdf.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2014.

Kiraly, F. J. and Oberhauser, H. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20(31):1–45, 2019. URL http://jmlr.org/papers/v20/16-314.html.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. Inter-domain Gaussian processes for sparse inference using inducing features. In *Advances in Neural Information Processing Systems*, pp. 1087–1095, 2009.

Li, P., Hastie, T., and Church, K. Very sparse random projections. volume 2006, pp. 287–296, 01 2006. doi: 10.1145/1150402.1150436.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., and Watkins, C. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

Lyons, T. and Xu, W. Inversion of signature for paths of bounded variation, 2011.

Lyons, T. J., Caruana, M., and Lévy, T. Differential equations driven by rough paths, 2007. Lectures from the 34th Summer School on Probability Theory held in Saint-Flour, July 6–24, 2004, With an introduction concerning the Summer School by Jean Picard.

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

Matthews, A. G. d. G., Hensman, J., Turner, R., and Ghahramani, Z. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. *Journal of Machine Learning Research*, 51:231–239, 2016.

Mattos, C. L. C., Dai, Z., Damianou, A., Forth, J., Barreto, G. A., and Lawrence, N. D. Recurrent Gaussian processes. In Larochelle, H., Kingsbury, B., and Bengio, S. (eds.), *Proceedings of the International Conference on Learning Representations*, volume 3, Caribe Hotel, San Juan, PR, 00 2016. URL http://inverseprobability.com/publications/mattos-recurrent16.html.

McInnes, L., Healy, J., Saul, N., and Grossberger, L. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

Micchelli, C. A., Xu, Y., and Zhang, H. Universal kernels. *Journal of Machine Learning Research*, 7(Dec):2651–2667, 2006.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

Rasmussen, C. E. and Williams, C. K. Gaussian processes for machine learning. 2006. *The MIT Press, Cambridge, MA, USA*, 38:715–719, 2006.

Salimbeni, H., Dutordoir, V., Hensman, J., and Deisenroth, M. Deep Gaussian processes with importance-weighted variational inference. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5589–5598, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Saxe, A. M., Mcclelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *In International Conference on Learning Representations*, 2014.

Schäfer, P. and Leser, U. Multivariate time series classification with weasel+muse. *ArXiv*, abs/1711.11343, 2017.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pp. 1257–1264, 2006.

Sriperumbudur, B. K., Fukumizu, K., and Lanckriet, G. R. Universality, characteristic kernels and rkhs embedding of measures. *Journal of Machine Learning Research*, 12 (Jul):2389–2410, 2011.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Takens, F. Detecting strange attractors in turbulence. In *Dynamical systems and turbulence, Warwick 1980*, pp. 366–381. Springer, 1981.

Titsias, M. Variational learning of inducing variables in sparse gaussian processes. In van Dyk, D. and Welling, M. (eds.), *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pp. 567–574, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 16–18 Apr 2009. PMLR. URL http://proceedings.mlr.press/v5/titsias09a.html.

Tuncel, K. S. and Baydogan, M. G. Autoregressive forests for multivariate time series modeling. *Pattern Recognition*, 73:202–215, 2018.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. Convolutional gaussian processes, 2017.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 370–378, Cadiz, Spain, 09–11 May 2016. PMLR.