Deep Signatures

Patric Bonnier[1,*] **Patrick Kidger**[1,2,*] **Imanol Perez Arribas**[1,2,*]

Cristopher Salvi[1,2,*] **Terry Lyons**[1,2]

[1] Mathematical Institute, University of Oxford
[2] Alan Turing Institute
`{bonnier, kidger, perez, salvi, tlyons}@maths.ox.ac.uk`

Abstract

The signature of a stream of data is a collection of statistics known to provide an efficient description of the stream. It has previously been treated as a fixed feature transformation, on top of which a model may be built. But by mapping an input stream to a *stream of streams* in a learnable way, and then applying the signature map stream-wise, we obtain a *learned*, *stream of signatures*. We emphasise both parts: not only are the terms of the signature selected in a learned way, but because the result is also a stream then the whole operation may be repeated arbitrarily many times. When the learnable map is a neural network then the operation of the signature may be interpreted as an activation function not operating element-wise; it is exploiting the stream-like nature of the data. In this manner signatures may be embedded as a layer anywhere within a neural network. We present the results of empirical experiments to back up the theoretical justification. Code available at `github.com/patrick-kidger/Deep-Signatures`.

1 Introduction

When data is ordered sequentially then it comes with a natural path-like structure: the data may be thought of as a discretisation of a path $X\colon [0,1] \to V$, where V is some Banach space. In practice we shall always take $V = \mathbb{R}^d$ for some $d \in \mathbb{N}$. For example the changing air pressure at a particular location may be thought of as a path in $\mathbb{R}$; the motion of a pen on paper may be thought of as a path in $\mathbb{R}^2$; the changes within financial markets may be thought of as a path in $\mathbb{R}^d$, with d potentially very large.

Given a path, we may define its *signature*, which is a collection of statistics of the path.

Definition 1.1. Let $\mathbf{x} = (x_1, \ldots, x_n)$, where $x_i \in \mathbb{R}^d$. Let $X\colon [0,1] \to \mathbb{R}^d$ be continuous, such that $X_{(i-1)/(n-1)} = x_i$, and linear on the intervals in between. Then letting $\otimes$ denote the tensor product, the signature of $\mathbf{x}$ is defined as the collection of iterated integrals

$$\begin{aligned}\mathrm{Sig}(\mathbf{x}) &= \left(\int_{0<t_1<\cdots<t_k<1} \mathrm{d}X_{t_1} \otimes \cdots \otimes \mathrm{d}X_{t_k}\right)_{k\geq 0} \\ &= \left(\left(\int_{0<t_1<\cdots<t_k<1} \mathrm{d}X^{i_1}_{t_1} \cdots \mathrm{d}X^{i_k}_{t_k}\right)_{1\leq i_1,\ldots,i_k\leq d}\right)_{k\geq 0}.\end{aligned}$$

We refer the reader to [1] for a primer on the use of the signature in machine learning. A brief overview of its key properties may be found in Appendix A. In short, the signature of a path

[*]Equal contribution.

Preprint. Work in progress.

determines the path essentially uniquely, and does so in an efficient, computable way. Furthermore, the signature is rich enough that every continuous function of the path may be approximated arbitrarily well by a linear function of its signature. Taken together these properties make the signature an attractive tool for machine learning. Most simply, the signature may be used as feature transformation, as it may often be simpler to learn a function of the signature than of the original path.

Originally introduced and studied by Chen in [2, 3, 4], the signature has seen use in finance [**?**, **?**], rough path theory [5, 6] and machine learning [7, 8, 9, 10, 11, 12, 13, 14, 15].

The signature is an infinite sequence, so in practice some finite collection of terms must be selected. Since the magnitude of the terms exhibit factorial decay, see Appendix A, it is usual [**?**] to simply choose the first N terms of this sequence, which will typically be the largest terms. These first N terms are called the *truncated signature*, denoted Sig^N. But if the function to be learned depended nontrivially on the higher degree terms, then crucial information has nonetheless been lost.

This may be remedied. Apply a pointwise augmentation to the original stream of data before taking the signature. Then the first N terms of the signature may better encode the necessary information [14, 15]. Explicitly, let $\Phi \colon \mathbb{R}^d \to \mathbb{R}^e$ be fixed; one could ensure that information is not lost by taking $\Phi(x) = (x, \varphi(x))$ for some φ. Then rather than taking the signature of $\mathbf{x} = (x_1, \ldots, x_n)$, where $x_i \in \mathbb{R}^d$, instead take the signature of $\Phi(\mathbf{x}) = (\Phi(x_1), \ldots, \Phi(x_n))$. In this way one may capture higher order information from the stream in the lower degree terms of the signature.

1.1 Our work

But how should this augmentation Φ be chosen? Previous work has fixed it arbitrarily, or experimented with several options before choosing one [14, 15]. Observe that in each case the map $\mathbf{x} \mapsto \mathrm{Sig}^N(\Phi(\mathbf{x}))$ is still ultimately just a feature transformation on top of which a model is learnt. Our more general approach is to allow the selection of Φ to be data-dependent, by having it be learnt; in particular it may be a neural network. Furthermore there is no reason it should necessarily operate pointwise, nor (since it is now learnt) need it be of the form $(x, \varphi(x))$. In this way we may enjoy the benefits of using signatures while avoiding their main limitation.

But this means that the signature is essentially operating as a layer within a neural network. It consumes a tensor of shape (b, d, s) – corresponding to a batch of size b of paths in $\mathbb{R}^d$ that have been sampled s times – and returns a tensor of shape $(b, (d^{N+1} - 1)/(d - 1))$, where N is the number of terms used in the truncated signature.[2] The signature is essentially an activation function that does not operate element-wise.

There is no reason to stop here. If the signature layer works well once then it is natural to seek to use it again. The obvious problem is that the signature map consumes a stream of data and returns statistics which have no obvious stream-like qualities. The solution is to lift the input stream to a *stream of streams*; for example, the stream of data $(x_1, \ldots, x_n)$ may be lifted to the 'expanding snapshots' of $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_i = (x_1, \ldots, x_i)$. Now apply the signature to each stream to obtain a stream of signatures $(\mathrm{Sig}^N(\mathbf{x}_1), \ldots, \mathrm{Sig}^N(\mathbf{x}_n))$, which is essentially a stream in Euclidean space. And now, on this new stream, we may augment it via a neural network and repeat the process again, as many times as we wish.

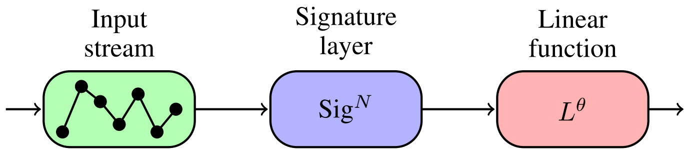
In this way we arrive at the notion of *deep signatures*.

The remainder of the paper is laid out as follows. In Section 2 we briefly discuss some related work, in Section 3 we detail the specifics of embedding the signature as a layer within a neural network. Section 4, 5, 6, and 7 are dedicated to experiments; we demonstrate positive results in supervised learning problems, generative problems, and reinforcement learning problems.

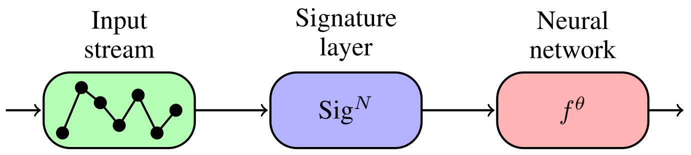
2 Related Work

Some related work is by necessity already discussed in the introduction. We expand a little on their proposed models here.

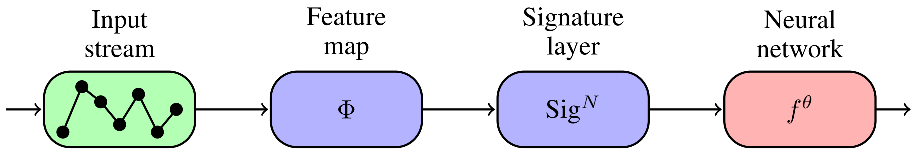
[2]Observe that $(d^{N+1} - 1)/(d - 1) = \sum_{k=0}^{N} d^k$ is the number of scalar values in a signature with N terms.



(a) Linear-signature model. Trainable parameters: θ.



(b) Neural-signature model. Trainable parameters: θ.



(c) Neural-signature-augment model. Trainable parameters: θ.

Figure 1: Three simple architectures with a signature layer.

Definition 2.1. The space of streams of data is defined as

$$\mathcal{S}(\mathbb{R}^d) = \{\mathbf{x} = (x_1, \ldots, x_n) : x_i \in \mathbb{R}^d, n \in \mathbb{N}\}.$$

Given $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$, the integer n is called the length of $\mathbf{x}$.

Three simple models utilising the signature layer are shown in Figure 1. In principle the universal approximation theorem for signatures (see Proposition A.5) guarantees that the model shown in Figure 1a is rich enough to learn any continuous function. In practice, of course, the signature must be truncated, and it is not clear how to appropriately choose the truncation hyperparameter N.

Thus a more practical approach is to replace the linear function on the signature by a nonlinear function, for example a neural network, see Figure 1b. This approach has been applied successfully in various tasks [7, 8, 9, 10, 11, 12, 13].

Following [14, 15], one could also apply a pointwise transformation to the stream before the signature, lifting the d-dimensional stream of data $(x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$ into a higher-dimensional feature space via a feature map $\Phi\colon \mathbb{R}^d \to \mathbb{R}^e$, yielding $(\Phi(x_1), \ldots, \Phi(x_n)) \in \mathcal{S}(\mathbb{R}^e)$. Then the signature of $\Phi(\mathbf{x})$ may potentially capture properties of the stream of data that will make learning from data easier, see Figure 1c.

3 The signature as a layer in a neural network

However, there is not always a clear candidate for the feature map Φ and a good choice might be very data-dependent. Thus we do the natural thing, and make Φ *learnable* by taking $\Phi = \Phi^\theta$ to be a neural network with trainable parameters θ. In this case, we again obtain the neural network shown in Figure 1c, except that Φ is now also learnable.

3.1 Stream-preserving neural networks

Recall that the signature consumes a tensor of shape (b, d, s) – corresponding to a batch of size b of paths in $\mathbb{R}^d$ that have been sampled s times – and returns a tensor of shape $(b, (d^{N+1} - 1)/(d-1))$, where N is the number of terms used in the truncated signature.

Let $(x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$ and $\Phi^\theta\colon \mathbb{R}^d \to \mathbb{R}^e$. Whatever the choice of Φ^θ, it must preserve the stream-like nature of the data if we are to take a signature afterwards. The simplest way of doing this is to have Φ^θ operate pointwise:

$$(\Phi^\theta(x_1), \ldots, \Phi^\theta(x_n)) \in \mathcal{S}(\mathbb{R}^e).$$

Another way to preserve the stream-like nature is to sweep a one dimensional convolution along the stream; more generally one could sweep a whole feedforward network along the stream. For some $m \in \mathbb{N}$ this gives

$$(\Phi^\theta(x_1, \ldots, x_m), \ldots, \Phi^\theta(x_{n-m+1}, \ldots, x_n)) \in \mathcal{S}(\mathbb{R}^e).$$

More generally still, this could have memory, as in a recurrent neural network. Fix $m \in \mathbb{N}$, let $\Phi_0 = 0$, and define $\Phi_k = \Phi^\theta(x_k, \ldots, x_{k+m}; \Phi_{k-1})$ for $k \in \{1, \ldots, n-m+1\}$. Then

$$(\Phi_1, \ldots, \Phi_{n-m+1}) \in \mathcal{S}(\mathbb{R}^e).$$

It is worth taking a moment to think what we really mean by 'stream-like nature'. The signature map is defined on paths; it is applied to a stream of data in $\mathcal{S}(\mathbb{R}^d)$ by first interpolating the data into a path and then taking the signature. That is, the data is treated as a discretisation or set of observations of some underlying path.

In principle one could reshape a tensor of shape (b, ds) with no stream-like nature into one of shape (b, d, s), and then take the signature. However it is not clear what this means mathematically: there is no underlying path which is being discretised. The signature is at this point an essentially arbitrary transformation, without the mathematical guarantees normally associated with it.

3.2 Backpropagation

At first glance the signature is not a pleasant-looking transformation. However despite being formed of integrals, the signature is in fact straightforward and efficient to compute, see Appendix A. More than that, the computation may in fact be described in terms of standard tensor operations. As such it may be backpropagated through without difficulty.

In particular this means that having a learnable map before the signature does not pose a problem.

3.3 Multiple signature layers

Now we would like not to restrict ourselves to a single signature layer. But as previously discussed, applying the signature layer 'consumes' the stream-like nature of the data. The straightforward solution is to take a stream of signatures in the following way: given a stream $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$, let $\mathbf{x}_k = (x_1, \ldots, x_k)$ for $k = 1, \ldots, n$, and apply the signature to each $\mathbf{x}_k$ to obtain the stream

$$(\mathrm{Sig}^N(\mathbf{x}_1), \mathrm{Sig}^N(\mathbf{x}_2), \ldots, \mathrm{Sig}^N(\mathbf{x}_n)) \in \mathcal{S}(\mathbb{R}^{(d^{N+1}-1)/(d-1)}).$$

Recall that the output of the signature is invariant to the length of the input stream.

This notion may be generalised: let

$$\ell = (\ell^1, \ell^2, \ldots, \ell^v) \colon \mathcal{S}(\mathbb{R}^d) \to \mathcal{S}(\mathcal{S}(\mathbb{R}^e)) \in \mathcal{S}(\mathbb{R}^{(e^{N+1}-1)/(e-1)}),$$

which we refer to as a lift into the space of streams of streams.[3] Then we apply the signature stream-wise to obtain

$$\mathrm{Sig}^N(\ell(\mathbf{x})) = \left(\mathrm{Sig}^N(\ell^1(\mathbf{x})), \ldots, \mathrm{Sig}^N(\ell^v(\mathbf{x}))\right).$$

In the above example, $\ell(\mathbf{x}) = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$. Other plausible choices are to cut up $\mathbf{x}$ into multiple pieces, for example

$$\ell(\mathbf{x}) = ((x_1, x_2), (x_3, x_4), \ldots, (x_{2\lfloor n/2 \rfloor - 1}, x_{2\lfloor n/2 \rfloor})),$$

or to take a sliding window

$$\ell(\mathbf{x}) = ((x_1, x_2, x_3), (x_2, x_3, x_4), \ldots, (x_{n-2}, x_{n-1}, x_n)).$$

And now we may repeat the whole process as many times as desired; see Figure 2. This defines the *deep signature model*. The name comes from the fact that the signature is now embedded deep within the network instead of operating as a feature transformation. When there are multiple signature layers in a network, so that signatures-of-signatures are computed, then the name may be taken to have a double meaning, but we stress that this need not be the case. Good results may often be obtained with only a single signature layer.

4 Supervised learning with fractional Brownian motion

[3] And v will likely depend on the length of the input to ℓ.

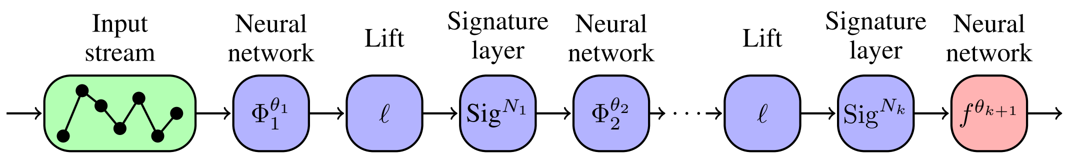
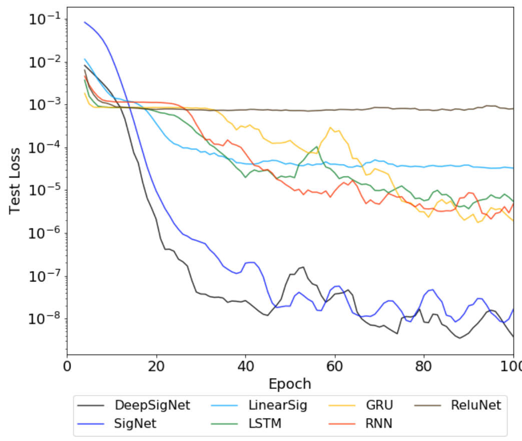


Figure 2: Deep signature model. Trainable parameters: $\theta_1, \ldots, \theta_{k+1}$.



(a) Test loss by epoch

	Test Loss	# Params
Rescaled Range	0.0032	N/A
Feedforward	0.0471	11889
RNN	0.0088	10635
GRU	0.0047	9780
LSTM	0.0039	10832
Linear-Sig	0.0112	10313
SigNet	0.0009	8777
DeepSigNet	**0.0002**	9741

(b) Final test loss

Figure 4: Performance at estimating the Hurst parameter for various models with and without signatures.

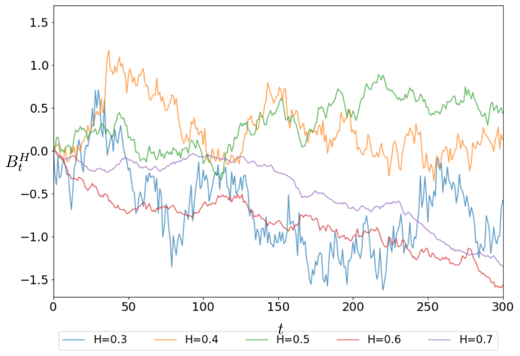


Figure 3: Realisations of fractional Brownian motion with different Hurst parameters.

Fractional Brownian motion [16] is a Gaussian process $B^H \colon [0, \infty) \to \mathbb{R}$ that generalises Brownian motion. It is self-similar and exhibits fractal-like behaviour. Unlike classical Brownian motion, increments of fractional Brownian motion need not be independent. Fractional Brownian motion depends upon a parameter $H \in (0, 1)$, known as the Hurst parameter. Examples are shown in Figure 3; note how lower Hurst parameters result in noticeably rougher paths.

Fractional Brownian motion has been successfully used to model phenomena in diverse fields. For example, empirical evidence from financial markets [17] suggests that log-volatility is well modelled by fractional Brownian motion with Hurst parameter $H \approx 0.1$. The case of $H = \frac{1}{2}$ corresponds to usual Brownian motion.

Estimating the Hurst parameter of a fractional Brownian motion path is considered a challenging task because of the paths' non-stationarity and long range dependencies [18]. We train a variety of models to perform this estimation. That is, to learn the map

$$\mathbf{x}^H \mapsto H,$$

where $\mathbf{x}^H = ((t_0, x_0), (t_1, x_1), \ldots, (t_n, x_n)) \in \mathcal{S}(\mathbb{R}^2)$ with $x_i = B^H_{t_i}$ for some realisation of B^H.

The results are shown in Figures 4 and 5.

All models were trained for the same number of epochs using the Adam [**?**] optimiser as implemented by PyTorch [**?**], which was the framework used to implement the models. Also shown

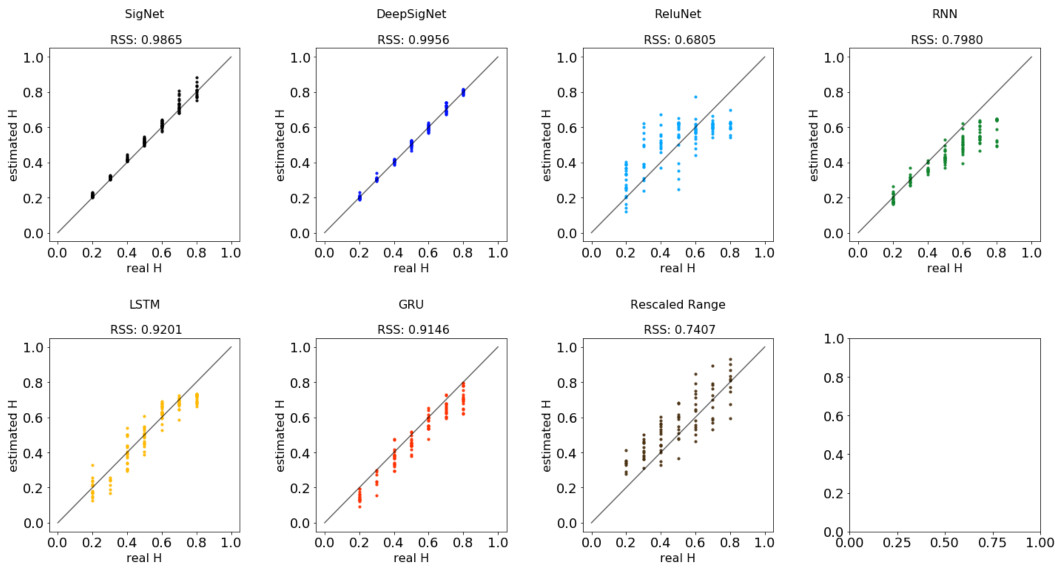


Figure 5: Real vs predicted Hurst parameters.

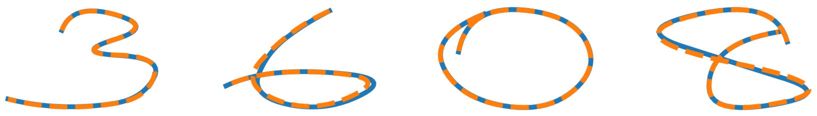


Figure 6: Original path (blue) and path reconstructed from its signature (dashed orange) for four handwritten digits in the PenDigits dataset [22].

are the results of the rescaled range method [**?**]. The data was also flattened and fed into a non-recurrent feedforward model, referred to as Feedforward, as a baseline in the context of neural networks, whilst the naïve Linear-Sig model outlined previously in Figure 1a provides a baseline from the context of signatures. SigNet and DeepSigNet are both models of the sort described by Figure 2; SigNet has a single Neural-Lift-Signature block, whilst DeepSigNet has two. TODO: Say a little more about the architectures. (size activation etc.)

5 Inverting the truncated signature

Given a truncated signature, we might seek to recover the original stream of data. The signature map is essentially injective (up to tree-like equivalence, see Appendix A), so this is a well-posed problem in principle. One immediate hurdle is that in practice we have the truncated signature rather than the full signature. Even given the full signature, finding a mathematical description of the inversion is a challenging task [19, 20, 21].

In this experiment we show how the signature layer can be used to perform this task. Fix a stream of data $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$. Assume that the truncated signature $\mathrm{Sig}^N(\mathbf{x})$ and the number of steps $n \in \mathbb{N}$ are known; we seek to reconstruct the stream of data $\mathbf{x}$. The strategy now is to minimise the following loss function, using backpropagation:

$$L(\mathbf{y}; \mathbf{x}) = \frac{1}{2} \left\| \mathrm{Sig}^N(\mathbf{y}) - \mathrm{Sig}^N(\mathbf{x}) \right\|_2^2 \quad \text{for } \mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{S}(\mathbb{R}^d).$$

Figure 6 shows four handwritten digits from the PenDigits dataset [22], for which $n =$ TODO. The solid blue path is the original path $\mathbf{x}$, whilst the dashed orange path is the reconstructed path $\mathbf{y}$ minimising $L(\mathbf{y}; \mathbf{x})$. Truncated signatures of order $N = 12$ were used for this task. We see that the truncated signatures have managed to encode the input paths $\mathbf{x}$ almost perfectly.

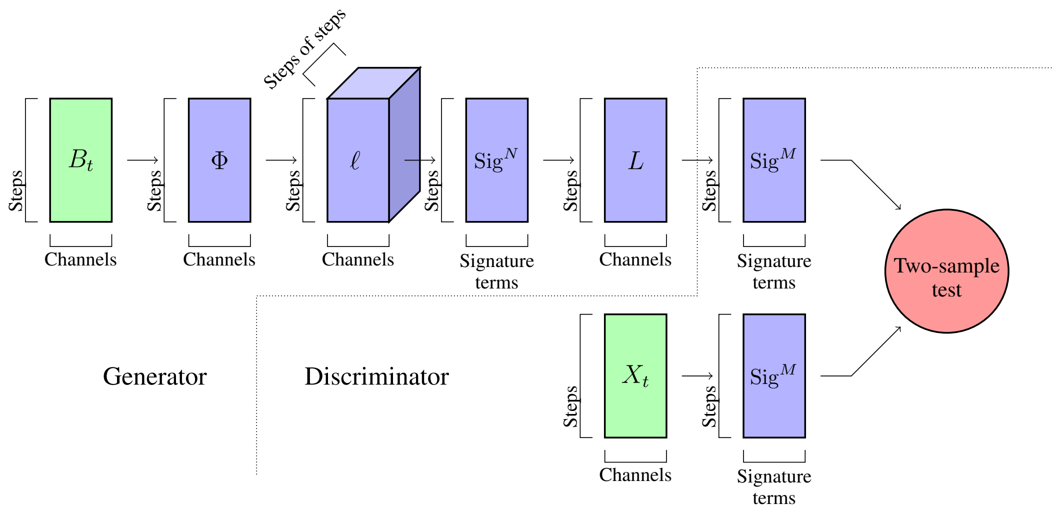


Figure 7: Generative model architecture

6 A generative model for a stochastic process

Generative models are typically trained to learn to transform random noise to a target distribution. One common approach are Generative Adversarial Networks (GANs) [23]. An alternative approach is to define a distance on the space of distributions by embedding them into a Reproducing Kernel Hilbert Space (RKHS). The discriminator is then a fixed two-sample test based on a kernel maximum mean discrepancy [24, 25, 26]. This is known as a Generative Moment Matching Network (GMMN).

With this framework we propose a signature-based generative model for sequential data.

Following [14, 15], the kernel k is given by

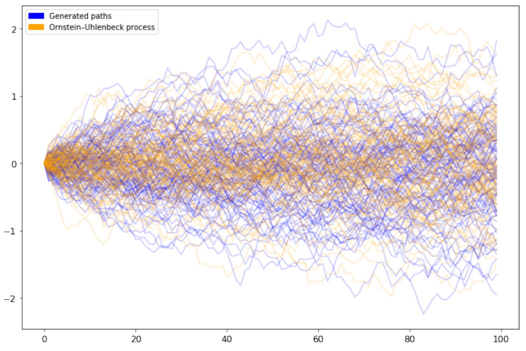


Figure 8: Generated paths alongside the original paths.

$$k\colon \mathcal{S}(\mathbb{R}^d)\times\mathcal{S}(\mathbb{R}^d) \to \mathbb{R}, \quad k(\mathbf{x},\mathbf{y}) = \left(\mathrm{Sig}^N\left(\lambda_{\mathbf{x}}\mathbf{x}\right), \mathrm{Sig}^N\left(\lambda_{\mathbf{y}}\mathbf{y}\right)\right),$$

where $\lambda_{\mathbf{x}} \in \mathbb{R}$ is a certain normalising constant guaranteeing that k is the kernel of a RKHS and $(\,\cdot\,,\,\cdot\,)$ denotes the dot product. Then given m samples $\{\mathbf{y}^{(i)}\}_{i=1}^n \subseteq \mathcal{S}(\mathbb{R}^d)$ from the target distribution and n samples $\{\mathbf{x}^{(i)}\}_{i=1}^n \subseteq \mathcal{S}(\mathbb{R}^d)$ from the generator, the loss is given by

$$T\left(\{\mathbf{x}^{(i)}\}_{i=1}^n, \{\mathbf{y}^{(i)}\}_{i=1}^m\right) = \frac{1}{n^2}\sum_{i,j} k(\mathbf{x}^{(i)},\mathbf{x}^{(j)}) - \frac{2}{nm}\sum_{i,j} k(\mathbf{x}^{(i)},\mathbf{y}^{(j)}) + \frac{1}{m^2}\sum_{i,j} k(\mathbf{y}^{(i)},\mathbf{y}^{(j)}).$$

The proposed model is shown in Figure 7; there is an implicit batch dimension throughout. The input to the network is Brownian motion B_t. This is fed through a small neural network Φ and lifted to a stream of streams by ℓ. The truncated signature is then applied stream-wise, and finally a linear layer is placed on the end. In a nice twist, both the generator and the discriminator involve the signature. Observe how the generative part is a particular case of the general form proposed in Figure 2.

We applied the proposed generative model to a dataset of 1024 realisations of an Ornstein–Uhlenbeck process [27]. The layer Φ was taken to be a small neural network with 2 output neurons and one hidden layer of 8 neurons, and a ReLU activation function, operating pointwise on the stream $(\mathbf{B_t}, \mathbf{t}) \in S(\mathbb{R}^2)$. The signature was truncated at $N = 3$ and linearly mapped by L down to dimension 2. Figure 8 shows the generated paths alongside the original ones. Minimising the loss function T implies that the generated paths are almost indistinguishable from the real Ornstein–Uhlenbeck process.

7 Reinforcement Learning

Acknowledgements

PK was supported by the Engineering and Physical Sciences Research Council [EP/L015811/1]. TODO: Any other acknowledgements.

TODO: Tidy up bibliography: there's quite a lot of 'arXiv preprint arXiv'

References

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

[2] K. T. Chen, "Iterated integrals and exponential homomorphisms," *Proc. London Math. Soc, 4, 502-512*, 1954.

[3] K. T. Chen, "Integration of paths, geometric invariants and a generalized baker-hausdorff formula," *Ann. of Math. (2), 65:163—178*, 1957.

[4] K. T. Chen, "Integration of paths - a faithful representation of paths by non-commutative formal power series," *Trans. Amer. Math. Soc. 89 (1958), 395–407*, 1958.

[5] T. J. Lyons, "Differential equations driven by rough signals," *Revista Matemática Iberoamericana*, vol. 14, no. 2, pp. 215–310, 1998.

[6] P. K. Friz and N. B. Victoir, "Multidimensional stochastic processes as rough paths: theory and applications," *Cambridge University Press*, 2010.

[7] W. Yang, L. Jin, and M. Liu, "Chinese character-level writer identification using path signature feature, dropstroke and deep cnn," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 546–550, IEEE, 2015.

[8] Z. Xie, Z. Sun, L. Jin, H. Ni, and T. Lyons, "Learning spatial-semantic context with fully convolutional recurrent network for online handwritten chinese text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 8, pp. 1903–1917, 2018.

[9] W. Yang, L. Jin, D. Tao, Z. Xie, and Z. Feng, "Dropsample: A new training method to enhance deep convolutional neural networks for large-scale unconstrained handwritten chinese character recognition," *Pattern Recognition*, vol. 58, pp. 190–203, 2016.

[10] W. Yang, L. Jin, and M. Liu, "Deepwriterid: An end-to-end online text-independent writer identification system," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 45–53, 2016.

[11] C. Li, X. Zhang, and L. Jin, "Lpsnet: a novel log path signature feature based hand gesture recognition framework," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 631–639, 2017.

[12] W. Yang, T. Lyons, H. Ni, C. Schmid, L. Jin, and J. Chang, "Leveraging the path signature for skeleton-based human action recognition," *arXiv preprint arXiv:1707.03993*, 2017.

[13] W. Yang, L. Jin, H. Ni, and T. Lyons, "Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 4083–4088, IEEE, 2016.

[14] F. J. Király and H. Oberhauser, "Kernels for sequentially ordered data," *arXiv preprint arXiv:1601.08169*, 2016.

[15] I. Chevyrev and H. Oberhauser, "Signature moments to characterize laws of stochastic processes," *arXiv preprint arXiv:1810.10971*, 2018.

[16] I. S. Mishura, I. S. Mishura, J. S. Mišura, Y. Mishura, and Û. S. Mišura, *Stochastic calculus for fractional Brownian motion and related processes*, vol. 1929. Springer Science & Business Media, 2008.

[17] J. Gatheral, T. Jaisson, and M. Rosenbaum, "Volatility is rough," *arXiv preprint arXiv:1410.3394*, 2014.

[18] L. Lacasa, B. Luque, J. Luque, and J. C. Nuno, "The visibility graph: A new method for estimating the hurst exponent of fractional brownian motion," *EPL (Europhysics Letters)*, vol. 86, no. 3, p. 30001, 2009.

[19] T. Lyons and W. Xu, "Inverting the signature of a path," *arXiv preprint arXiv:1406.7833*, 2014.

[20] J. Chang, N. Duffield, H. Ni, W. Xu, *et al.*, "Signature inversion for monotone paths," *Electronic Communications in Probability*, vol. 22, 2017.

[21] J. Chang, *Effective algorithms for inverting the signature of a path*. PhD thesis, University of Oxford, 2018.

[22] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[24] Y. Li, K. Swersky, and R. Zemel, "Generative moment matching networks," *ICML*, 2015.

[25] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," *UAI*, 2015.

[26] A. Gretton, K. M. Borgwardt, M. Rasch, B. Scholkopf, and A. J. Smola, "A kernel method for the two-sample problem," *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[27] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical review*, vol. 36, no. 5, p. 823, 1930.

[28] B. M. Hambly and T. J. Lyons, "Uniqueness for the signature of a path of bounded variation and the reduced path group," *Annals of Mathematics*, vol. 171, no. 1, pp. 109–167, 2010.

[29] T. Lyons, "Rough paths, signatures and the modelling of functions on streams," *arXiv preprint arXiv:1405.4537*, 2014.

A A brief overview of signatures

A.1 Properties of signatures

Definition A.1 ([5])**.** Let $a, b \in \mathbb{R}$, and $X = (X^1, \ldots, X^d) \colon [a, b] \to \mathbb{R}^d$ be a continuous piecewise smooth path. The signature of X is then defined as the collection of iterated integrals

$$\begin{aligned}\mathrm{Sig}(X) &= \left(\int_{a<t_1<\cdots<t_k<b} \mathrm{d}X_{t_1} \otimes \cdots \otimes \mathrm{d}X_{t_k} \right)_{k \geq 0} \\ &= \left(\left(\int_{a<t_1<\cdots<t_k<b} \mathrm{d}X^{i_1}_{t_1} \cdots \mathrm{d}X^{i_k}_{t_k} \right)_{1 \leq i_1, \ldots, i_k \leq d} \right)_{k \geq 0},\end{aligned}$$

where $\otimes$ denotes the outer product, $\mathrm{d}X_t = \frac{\mathrm{d}X_t}{\mathrm{d}t}\mathrm{d}t$, and the $k = 0$ term is taken to be $1 \in \mathbb{R}$.

The (truncated) signature of order N of X is defined as

$$\mathrm{Sig}^N(X) = \left(\int_{a<t_1<\cdots<t_k<b} \mathrm{d}X_{t_1} \otimes \cdots \otimes \mathrm{d}X_{t_k} \right)_{0 \leq k \leq N}.$$

The signature may in fact be defined much more generally, on paths of merely bounded variation, see [5, 6], but the above definition suffices for our purposes. This broader theory is also the reason behind the notation $\mathrm{d}X_t$, which may be made sense of even when X is not continuous or differentiable.

Example A.2. Suppose $X \colon [a, b] \to \mathbb{R}^d$ is the linear interpolation of two points $x, y \in \mathbb{R}^d$, so that $X_t = x + \frac{t-a}{b-a}(y - x)$. Then its signature is just the collection of powers of its total increment:

$$\mathrm{Sig}(X) = \left(1, y - x, \frac{1}{2}(y - x)^{\otimes 2}, \frac{1}{6}(y - x)^{\otimes 3}, \ldots, \frac{1}{k!}(y - x)^{\otimes k}, \ldots \right).$$

Which is independent of a, b.

The signature exhibits three key properties that makes their use attractive when dealing with path-like data. First, a path is essentially defined by its signature. This means that essentially no information is lost when applying the signature map.

Proposition A.3 (Uniqueness of signature [28])**.** *Let $X \colon [a, b] \to \mathbb{R}^d$ be a continuous piecewise smooth path. Then $\mathrm{Sig}(X)$ uniquely determines X up to 'tree-like equivalence'.*

Next, the terms of the signature decay in size factorially. This means that little information is lost when truncating the signature.

Proposition A.4 (Factorial decay [5, Lemma 2.1.1])**.** *Let $X \colon [a, b] \to \mathbb{R}^d$ be a continuous piecewise smooth path. Then*

$$\left\| \int_{a<t_1<\cdots<t_n<b} dX_{t_1} \otimes \cdots \otimes dX_{t_n} \right\| \leq \frac{C(X)^N}{N!},$$

where $C(X)$ is a constant depending on X and $\| \cdot \|$ is any tensor norm on $(\mathbb{R}^d)^{\otimes n}$.

Finally, functions of the path are approximately linear on the signature. In some sense the signature may be thought of as a 'universal nonlinearity' on paths.

Proposition A.5 (Universal approximation theorem for signatures [29])**.** *Let F be a real-valued continuous function on continuous piecewise smooth paths and let $\mathcal{K}$ be a compact set of such paths.*[4] *Let $\varepsilon > 0$. Then there exists a linear functional L such that*

$$|F(X) - L(\mathrm{Sig}(X))| < \varepsilon \quad \forall\, X \in \mathcal{K}.$$

[4]Of course the definition of both continuity and compactness depend on the topology of the set of paths. See [29] for details.

A.2 Computing signatures

The signature is little more than a mathematical idealisation unless we can compute it. Fortunately, this is possible, and in an efficient manner too.

Observe that the signature of a path $X\colon [a,b] \to \mathbb{R}^d$ can be described as a sequence where the zeroth term is $1 \in (\mathbb{R}^d)^{\otimes 0} = \mathbb{R}$, the first term belongs to $\mathbb{R}^d$, the second term belongs to $\mathbb{R}^d \otimes \mathbb{R}^d$, and the kth term belongs to $(\mathbb{R}^d)^{\otimes k} = \mathbb{R}^d \otimes \cdots \otimes \mathbb{R}^d$, k times. With this description, the signature of a path naturally takes values in the tensor algebra:

Definition A.6. The tensor algebra of $\mathbb{R}^d$ is defined as

$$T((\mathbb{R}^d)) = \prod_{k=0}^{\infty} (\mathbb{R}^d)^{\otimes k}.$$

The tensor product, when extended by bilinearity, naturally defines a multiplication on $T((\mathbb{R}^d))$. For $A = (A_0, A_1, \ldots) \in T((\mathbb{R}^d))$ and $B = (B_0, B_1, \ldots) \in T((\mathbb{R}^d))$, then $A \otimes B \in T((\mathbb{R}^d))$ can be seen to be

$$A \otimes B = \left(\sum_{j=0}^{k} A_j \otimes B_{k-j} \right)_{k \geq 0}.$$

A fundamental insight of Chen is that concatenation of paths corresponds to multiplication of their signatures. The following relation is known as *Chen's identity*.

Proposition A.7 (Chen's identity, [5, Theorem 2.12])**.** *Let $X\colon [a,b] \to \mathbb{R}^d$ and $Y\colon [a,b] \to \mathbb{R}^d$ be two continuous piecewise smooth paths such that $X_b = Y_a$. Define their concatenation $X * Y$ as*

$$(X * Y)_t = \begin{cases} X_{2t-a} & \text{for} \quad a \leq t < \dfrac{a+b}{2}, \\ Y_{2t-b} & \text{for} \quad \dfrac{a+b}{2} \leq t \leq b. \end{cases}$$

Then

$$\mathrm{Sig}(X * Y) = \mathrm{Sig}(X) \otimes \mathrm{Sig}(Y).$$

Everything so far has operated on paths. In practice, of course, one has some stream of data, which we interpret as a discretisation of a path.

Definition A.8. The space of streams of data is defined as

$$\mathcal{S}(\mathbb{R}^d) = \{\mathbf{x} = (x_1, \ldots, x_n) : x_i \in \mathbb{R}^d, n \in \mathbb{N}\}.$$

Given $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$, the integer n is called the length of $\mathbf{x}$. Furthermore for $a, b \in \mathbb{R}$ such that $a < b$, fix

$$a = u_1 < u_2 < \cdots < u_{n-1} < u_n = b.$$

Let $X = (X^1, \ldots, X^d)\colon [a,b] \to \mathbb{R}^d$ be continuous such that $X_{u_i} = x_i$ for all i, and linear on the intervals in between. Then X is called a linear interpolation of $\mathbf{x}$.

Definition A.9. Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{S}(\mathbb{R}^d)$ be a stream of data. Let X be a linear interpolation of $\mathbf{x}$. Then the signature of $\mathbf{x}$ is defined as

$$\mathrm{Sig}(\mathbf{x}) = \mathrm{Sig}(X)$$

and the (truncated) signature of order N of $\mathbf{x}$ is defined as

$$\mathrm{Sig}^N(\mathbf{x}) = \mathrm{Sig}^N(X).$$

A priori this definition of the signature of a stream of data depends on the choice of linear interpolation. However, it turns out that Definition A.9 is well-defined and independent of this choice, see [5, Lemma 2.12].

Remark A.10. Let $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{S}(\mathbb{R}^d)$ be a stream of data of length n in $\mathbb{R}^d$. Then $\mathrm{Sig}^N(\mathbf{x})$ has

$$\sum_{k=0}^{N} d^k = \frac{d^{N+1} - 1}{d - 1}$$

components. In particular, the number of components does not depend on n; the truncated signature maps the infinite-dimensional space of streams of data $\mathcal{S}(\mathbb{R}^d)$ into a finite-dimensional space of dimension $(d^{N+1} - 1)/(d - 1)$. Thus the signature is an excellent way to tackle long streams of data, or streams of variable length.

Equipped with Chen's identity, the signature of a stream is straightforward to compute explicitly.

Proposition A.11. *Let* $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{S}(\mathbb{R}^d)$ *be a stream of data. Then,*

$$\mathrm{Sig}(\mathbf{x}) = \exp(x_2 - x_1) \otimes \exp(x_3 - x_2) \otimes \cdots \otimes \exp(x_n - x_{n-1}),$$

where

$$\exp(x) = \left(\frac{x^{\otimes k}}{k!} \right)_{k \geq 0}.$$

Proof. It is easy to check that if $\mathbf{x} = (x_1, x_2) \in \mathcal{S}(\mathbb{R}^d)$ is a stream of data of length 2 then the signature of $\mathbf{x}$ is given by $\exp(x_2 - x_1)$, as in Example A.2. So given a stream of data $\mathbf{x} = (x_1, \dots, x_n) \in \mathcal{S}(\mathbb{R}^d)$ of length $n \geq 2$, iteratively applying Chen's identity yields the result. $\square$

Remark A.12. Chen's identity implies that computing the signature of an incoming stream of data is efficient. Indeed, suppose one has obtained a stream of data $\mathbf{x} \in \mathcal{S}(\mathbb{R}^d)$ and computed its signature. Suppose that after some time more data has arrived, $\mathbf{y} \in \mathcal{S}(\mathbb{R}^d)$. In order to compute the signature of the whole signal one only needs to compute the signature of the new piece of information, and tensor product it with the already-computed signature.

Remark A.13. The signature has one final property, which we have actually already briefly touched upon, which is that it is invariant to reparameterisations. Formally speaking, if $X \colon [0, 1] \to \mathbb{R}^d$ is a continuous path, and $\psi \colon [0, 1] \to [0, 1]$ is an orientation-preserving diffeomorphism, then $\mathrm{Sig}(X) = \mathrm{Sig}(X \circ \psi)$. It is the reason why the signature of a stream of data is invariant to the choice of linear interpolation. Thus the signature encodes the *order* in which data arrives without caring precisely *when* it arrives. For example, consider the scenario of recording the movement of a pen as it draws a character on a piece of paper. Then the signature of the stream of data is invariant to the speed at which the character was drawn.