Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

UNIVERSAL WISER PUBLISHER

Research Article

Signature Transform Application in Time Series Analysis: An Introduction to Classification and Reconstruction Problems

Biagio Paparella[1*], **Francesco Grimaccia**[2]

[1] Institute for Numerical Simulation, University of Bonn, Friedrich-Hirzebruch-Allee 7, Bonn, 53115, Germany
[2] Department of Energy, Polytechnic University of Milan, Leonardo Da Vinci Square, 32, Milan, 20133, Italy
E-mail: b.paparella@rivacon.com

Received: 31 October 2024; **Revised:** 13 January 2025; **Accepted:** 13 January 2025

Abstract: The goal of this paper is to analyze comprehensively how the *Signature Transform* can be used for the classification of time series. Data preprocessed with the signature transform can theoretically be classified using only *linear* models, which represents an advantage over more traditional algorithms. Unlike existing papers, our goal is to offer a self-contained, compact work that is accessible to audiences with a more limited mathematical background. The exposition is deliberately soft, and the experiments are short and clear, aiming to provide all the details necessary for full reproducibility. In our illustrative example, we work with sine waves (signals) of only two different possible frequencies, carefully pointing out every step of the methodology. Remarks on how to visualize time series in a simple two dimensional (2D) plane are included, as visualization represents an important step in developing an intuitive understanding of the problem. Finally, we discuss an alternative application in signal reconstruction, with the aim of building an encoder/decoder system. Important limitations arise in this final section, which nevertheless open a way to improve a clear and intuitive interpretation of such a transform.

Keywords: signature transform, time series analysis, signals classification

MSC: 62M10, 68T10

1. Introduction

The *signature transform* is an important mathematical object coming from the field of stochastic analysis [1] and controlled differential equations [2]. In recent years, there has been a growing interest in using these techniques in problems concerning statistics and machine learning, as explained in the gentle introduction [3] or in the more technically extended work [4]. Applications in generative learning can be found in [5]. They all agree on the intuition that one of the main reasons for the success of the signature lies in its ability to "capture the geometry of a path", making it a potentially interesting tool.

From a more traditional perspective, this transform connects to solutions of differential equations emerging from the use of Picard methods and iterative integrals. The idea can be generalized and finally detached from its original context: the signature becomes an intrinsic object of any *path* itself, independent of possible connections to differential equations. Since paths (from a mathematical viewpoint) can be interpreted as "time series" in the language of statistics, the two

DOI: https://doi.org/10.37256/cm.6220256004

worlds can be connected. This remark is very important since analyzing time series is of crucial importance in different fields ranging from economics to medicine and engineering.

In our studies, we suppose we have a dataset $\mathcal{D} = \{\hat{x}_n, y_n\}_{n=1,\dots,N}$ of N samples, such that $\hat{x}_n$ are arrays of collected values during measurements, while the scalars y_n are *labels* that can be either 0 or 1. For instance, $\hat{x}_n$ can represent the temperature oscillations during the n-th day, with y_n set to zero or one according to the presence of a meteorological phenomenon of interest on that day. In medical applications, $\hat{x}_n$ can be the heart rate of a patient, with y_n being a flag that detects a pathology. In a completely different context, the measurements can be stock values, with the labels used to indicate if there is a trend of interest.

We allow the measurements $\hat{x}_n$ to have different lengths. This makes the use of standard algorithms less straightforward, since, for instance, pointwise comparisons are no longer possible. As is common in statistics and machine learning, the final goal is to construct a model capable of determining whether a time series belongs to the class with label 0 or 1, as described in [6]. We propose the following methodology:

(i) Every sample $\hat{x}_n$, which is interpreted as a time series, is **compressed** into a new array $\hat{p}_n$, its (approximated) signature. Because of properties explained later, the arrays $\hat{p}_n$ all have the same length. A new dataset $\mathcal{P} = \{\hat{p}_n, y_n\}_{n=1,\dots,N}$ is constructed. In this step, the labels do not play any role; the signature only involves the time series $\hat{x}_n$;

(ii) On the new dataset $\mathcal{P}$, now composed of elements of the same shape, a simple **linear map** is used to perform the classification according to the labels y_n;

(iii) To improve the understanding of the entire dataset, the dataset $\mathcal{P}$ is plotted in two dimensions using the *Multidimensional Scaling Algorithm* (abbreviated as MDS, see [7]) to **visualize** all the (compressed) time series in a single readable figure. Alternative methods for this dimension reduction can be used (like t-SNE, PCA, or similar techniques), but we prefer the use of MDS for its isometry properties as better explained later.

In the following section, we introduce and define the signature transform, choosing a concise and self-contained approach that limits all possible technicalities. We proceed by adding more details about the methodology and carefully describing the numerical experiments with sinusoids of different frequencies. The paper concludes by commenting on the benefits and limits of the proposed methodology, including applications in signal reconstruction.

We plan to exploit our research with future work on prediction and statistical analysis, thereby extending the signature use for the most frequent use cases when working especially with time series data.

2. The signature transform

2.1 *Definition*

Let $f\colon [0, 1] \to \mathbb{R}$ be a piecewise linear function with $f(0) = 0$, intuitively representing the measurements composing the observed time series. For functions that do not start from 0, one should coherently preprocess the data (e.g., by applying a translation) to meet this condition in order to guarantee some mathematical properties later clarified. To keep track of time, we always consider paths *augmentations* $X(t) = (t, f(t))$. We denote $X^i(t)$ as the i-th coordinate of X, counting from 0, such that $X^0(t) = t$ and $X^1(t) = f(t)$. Before continuing, we need some preliminary combinatorics.

Definition 2.1 (Multi-index) A multi-index of depth $n \geq 1$ is a sequence of n digits, $(i_1, \dots, i_n)$, each digit being 0 or 1.

For instance, $(0, 1)$ is a multi-index of depth 2, while $(0, 0, 0, 1, 1)$ is of depth 5. For each depth n, we have a total of 2^n possible multi-indices. Therefore, if we consider all of them up to a certain depth n, we obtain a total of $\sum_{i=1}^{n} 2^i = 2^{n+1} - 2$ possible elements. At depth 4, we already have 30 coefficients: this rapid growth is summarized in Figure 1 (red dashed plot) and provides insight into the computational cost of the signature transform.

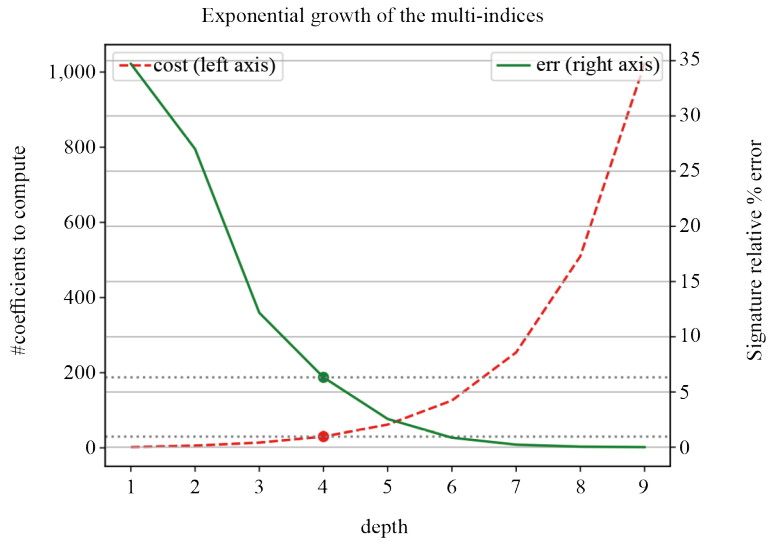

Figure 1. The exponential growth of the number of possible multi-indices increase (dashed red), and their corresponding empirical truncated signature error (solid green)

Definition 2.2 (Canonical simplex) The k-dimensional canonical simplex is the set:

$$\Delta^k = \{(u_1, \ldots, u_k), u_i \in [0, 1], 0 < u_1 < \cdots < u_k < 1\}.$$

Definition 2.3 (Signature coefficient) The signature coefficient of the path $X(t) = (t, f(t))$ corresponding to the multi-index $(i_1, \ldots, i_n)$ is the real number $s_{(i_1, \ldots, i_n)}$ defined as:

$$s_{(i_1, \ldots, i_n)} = \int_{\Delta^n} \dot{X}^{i_1}(u_1) \cdots \dot{X}^{i_n}(u_n) du_1 \cdots du_n, \tag{1}$$

where $\dot{X}^k$ is the derivative of the k-th component of the map X: either f' or the constant 1.

Definition 2.4 (The Signature Transform) The *signature transform* $S(X)$ of a path $X(t) = (t, f(t))$ is the infinite sequence of all the possible signature coefficients computed as above. When stopping at a certain depth N (included), the obtained sequence is called the *truncated signature* $S^N(X)$.

Since, in practice, we can only access finite sequences, we focus on truncated signatures. Figure 1 suggests how long any sequence of this type is expected to be. For instance, when considering $S^4(X)$, we obtain a collection of 30 real-number coefficients.

The truncation level should ultimately be seen as a hyperparameter to tune according to the needs, as is typical in Machine Learning practice, taking into account the performance results as well as the available computational power. On the other hand, there are mathematical results that support such a choice. We refer to [2] (Section 2, Proposition 2.2) for an actual formal exposition, and we quickly sketch the key intuition here. The absolute values of the signature coefficients can generally grow as the level increases, but there is always a certain depth after which an asymptotic decay to zero must start. The speed (rate) of such a phenomenon is governed by the length of the curve (more precisely, its 1-variation): the smaller the 1-variation, the sooner this decay starts. This is the reason why very small truncation levels (2 or 3) can already suffice when working with "short" curves, while highly oscillating data would generally require a longer truncation.

In our application, we initially computed signatures up to depth 10 and observed that most of the later coefficients were moving very close to zero, which empirically suggested that we were in this decaying regime. Therefore, we

considered the use of $N = 10$ to be acceptable. Then, we noticed that truncating at level 4 resulted in an average coefficient difference of 5% (compared to the full vector obtained at truncation 10), leading to the choice of $N = 4$ to save computational time. Various differences taken into account across multiple depths are displayed in the green plot of Figure 1.

Summing up, changing the truncation depth affects the overall performance of the algorithm: excessively long truncations slow down the computational process, while excessively short ones risk losing information about the time series (see *injectivity* in the paragraph below). Finding a proper trade-off between these two extremes is part of the implementation challenge.

To conclude this section, we invite the reader to consult Figure 2. A noisy wave from our dataset is represented, followed by its signature truncated at a depth of 4. The vertical bands separate the coefficients according to the depth to which they belong: the first contains 2 numbers, the next 4, then 8, and finally, the last includes 16 values, for a total of 30 coefficients.

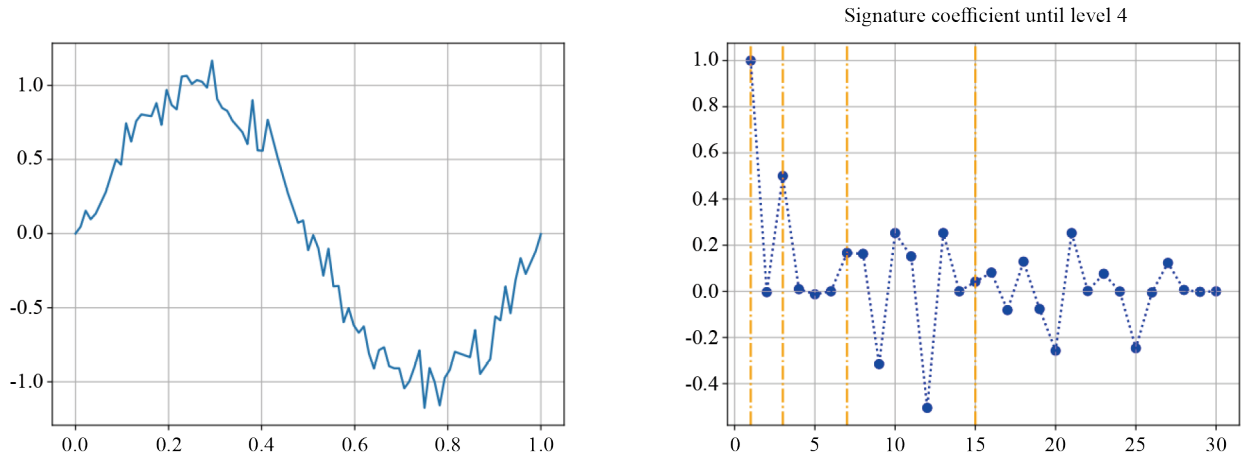

Figure 2. A noised wave with its signature until depth 4, composed by 30 coefficients

2.2 *Main properties*

When working with paths $X(t) = (t, f(t))$ starting from $(0, 0)$, the following properties are guaranteed:

1. **injectivity**: different time series always have different signatures (cf. [8], Proposition 1). On the other hand, since we only consider truncated signatures, this property is weakened. In principle, it is possible to have two different paths with identical signatures until a certain depth, differing only afterward. Unfortunately, we cannot prevent this possibility in advance;

2. **linearity**: any continuous functional operating *on* time series (for instance, a classifier operator), can be approximated by a *linear* function on the signatures of the data (cf. [8], Proposition 2);

3. **homogeneity**: directly by construction, even if two given time series have a different number of points, their truncated signatures always share the same length.

For the actual signature computations we rely on the Python library *Signatory* (https://pypi.org/project/signatory/). We recommend the reference paper [9] for more insights about the Signature computational complexity, algorithmic implementations and code examples.

3. Linear classifiers

Once the original measurements $\hat{x}_n$ are transformed using the signature, the remaining step consists of classification. Justified by the property briefly explained in the section above, we simply use a *linear classifier*: any affine map c: $\mathbb{R}^d \rightarrow$

$\mathbb{R}^2$ acting on d-dimensional data, with d being the length of the considered signature (e.g., 30 in the case of truncation at depth 4). The output position containing the highest value will determine the label assigned to the point. For instance, data with an output of (3.4, 12.1) is classified as 1, while data producing an output of (4.1, 0.34) is classified as 0. As usual, the goal is to tune c so that it manages to correctly assign the labels to a target set of data. The dataset is halved into two (a subdivision into three parts (training, validation, and testing) would be more realistic and aligned with modern practices; however, we kept the training/validation structure for a minimalist approach) random (balanced) subsets: the *train* part on which the actual optimization is performed, where the candidate function c is found, and the *validation* set (kept "hidden" during the tuning step) used later for model evaluation. The classification is considered a success if the function c is able to produce satisfactory accuracy (percentage of correct labels) on both the training and validation datasets.

4. Multidimensional scaling

Visualizing data is a procedure of great importance since it enhances intuitive understanding and can help when something is not working as expected. In this section, we explain a simple strategy that integrates well with the signature transform. Once the entire original dataset is transformed, the mutual differences between the obtained signatures inform us how "far" their corresponding time series are (due to the injectivity property previously explained). It is possible to store these values in a *distance matrix* of shape $N \times N$: this matrix is very informative; however, since its points are in high dimensions (for instance, 30 in the case of signatures truncated at depth 4), it is difficult to read.

On the other hand, the Multidimensional Scaling algorithm (abbreviated MDS, available in Python (https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html) and fully explained in the paper [7]) implements a simple but effective idea: given any distance matrix, it searches for points in dimension 2 such that their distance matrix is (approximately) the same as the original. Although this is not always possible, it is very easy to perform an integrity check: once points in dimension two are proposed, it is sufficient to compare their distance matrix to the original one and decide if the error is low enough to still convey meaningful information. By using this trick on the signature of our data, we are able to "see" them on the plane with consequential benefits. Additionally, the authors want to emphasize that it is certainly possible to use other dimensionality reduction techniques on the obtained signature data. We preferred and highlighted the application of MDS because it preserves the geometry (mutual distances) between them, improving the interpretation of such a transform operation.

5. Our experiment

Our time series are generated using simple mathematical equations in Figure 3. We construct a model capable of recognizing whether sine waves have lower (label 0) or higher (label 1) frequencies. This use case is purely illustrative. Readers are encouraged to build different time series datasets according to their needs. We generate a total of 300 time series, $X(t) = (t, f(t))$, with 150 paths governed by the equation $f_0(t) = \sin(2\pi t)$ (label 0) and the remaining 150 by $f_1(t) = \sin(4\pi t)$ (label 1). The time variable goes from 0 to 1, $t \in [0, 1]$, and this interval is uniformly discretized each time with a random number of subdivisions chosen between 60 and 100 steps. Around 10% of the points are then discarded, and the remaining points are perturbed (in their second component, f) by normal random variables with a mean of 0 and a standard deviation of 0.01 (simulating possible "losses" during realistic measurements).

Each time series is finally translated to have a value of $(0, 0)$ at time $t = 0$, which is important to theoretically guarantee all the signature properties previously mentioned.

We proceed by computing the signature for each path up to depth 4, obtaining a dataset $\mathcal{P}$ of shape 300×30 (300 samples of vectors of length 30), randomly split into 150 points for the training set and 150 for validation. On these sets, we check that a $50 : 50$ label proportion is still (approximately) respected.

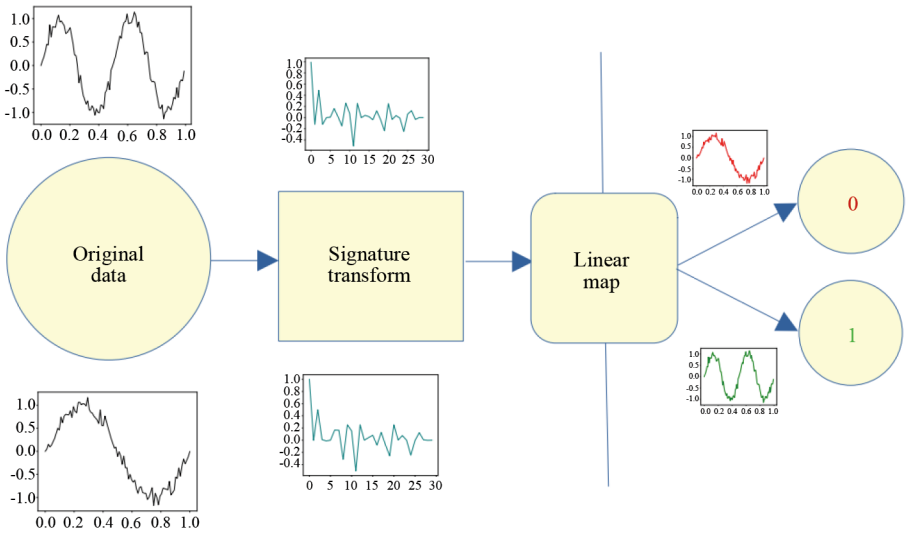

Figure 3. Time series are preprocessed with the signature and sent into a linear classifier

We use the PyTorch (https://pytorch.org/) library to perform the optimization task, using a classic Stochastic Gradient Descent (https://pytorch.org/docs/stable/generated/torch.nn.Linear.html, https://pytorch.org/docs/stable/generated/torch.optim.SGD.html) with learning rate $1e-3$ and running for 100 epochs. The chosen minimized loss function is the Cross Entropy, standard for classification tasks and available in the documentation (pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html).

Once the optimization step is concluded and the affine classifier m: $\mathbb{R}^{30} \to \mathbb{R}^2$ computed, the model m is evaluated on both the training and validation data and its accuracy measured. As explained in section 3, for any time series, evaluating m on its truncated signature produces a pair such that the coordinate with the highest value determines the label 0 or 1. The results are stored in Table 1 and point out the effectiveness of the proposed methodology, which overall required less than 30 seconds to execute on a modern laptop (without GPU acceleration).

Table 1. Results of our classification method

Training accuracy	Validation accuracy
96%	98%

By using Multidimensional Scaling, the entire dataset $\mathcal{P}$ is plotted in dimension 2 (Figure 4), making it visually clear how the data form two linearly separable clusters. Red dots correspond to (signatures of) time series with label 0, and green dots represent label 1. Blue dots indicate misclassified time series. Furthermore, regular triangles portray data in the training set, while inverted shapes are used for data in the validation set. The norm of the distance matrix for the points in 2D and that of the original dataset differ by less than 1%, therefore, we consider this MDS visualization reliable.

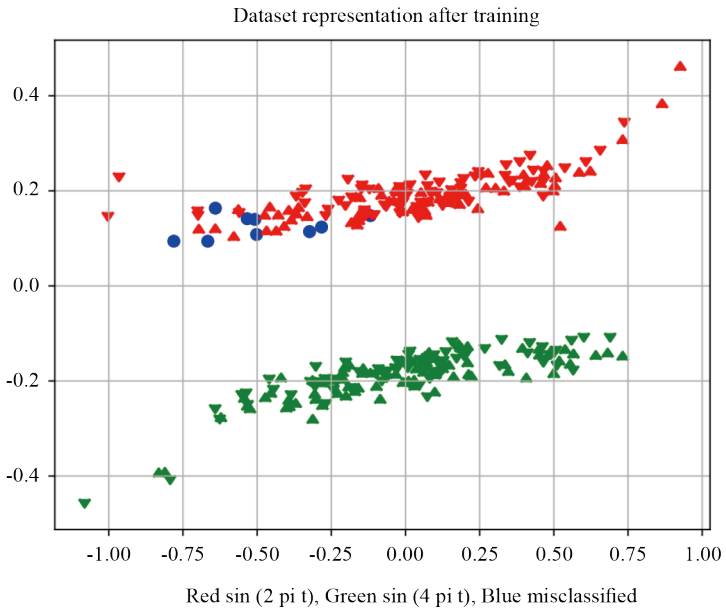

Figure 4. Dataset projected in dimension two by using MDS on the signature of data

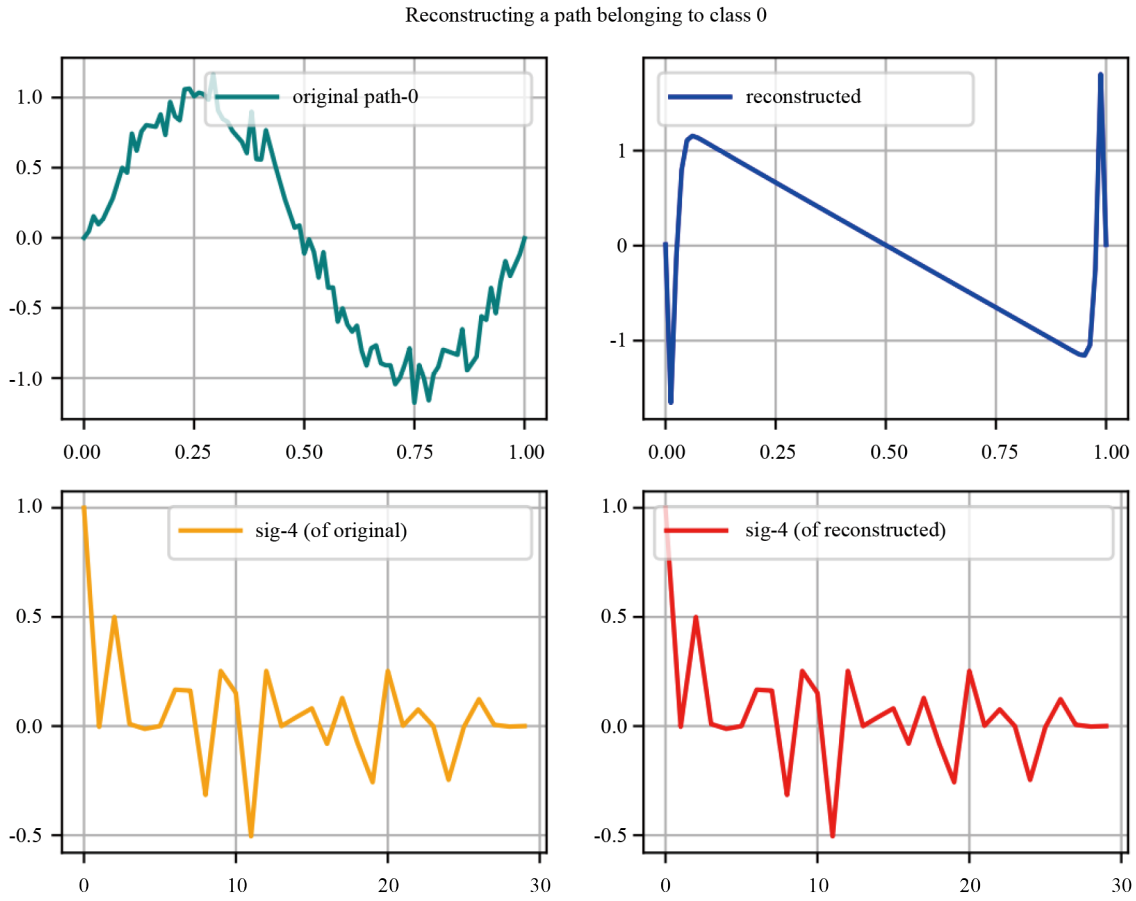

Figure 5. Reconstruction attempt for a noised single sine wave

6. Reversing the signature

This last section is actually independent of the previous classification problem. We would like to point out a possible further application and, at the same time, highlight some important limitations. We briefly discuss the problem of determining a time series, *given* its signature, with the idea of constructing a coder/encoder for the transmission of signals. Reversing the signature is generally considered to be a challenging problem [10, 11], and to our knowledge, there is no "definitive" answer in the literature. Despite the theoretical difficulties pointed out in the aforementioned literature for the general case, when working with one-dimensional time series composed of a very limited number of observations, we can, for instance, simply try an adaptive gradient descent approach, looking for time values such that the difference between their signature and the target one is minimized.

We followed this idea by exploiting PyTorch's automated differentiation and obtained mixed results worth commenting on, summed up in Figures 5, 6, and 7. Each of them is composed of four plots. In the top-left position, there is the original curve (green); below (bottom-left) are its 30 signature coefficients (in orange), shown as a continuous curve for readability. This is the signature provided as input for the reconstruction process. The reconstructed path is in blue and located in the top-right quadrant. Its 4-signature is then computed and displayed in the bottom-right position, in red, and is expected to be approximately the same as the original orange signature.

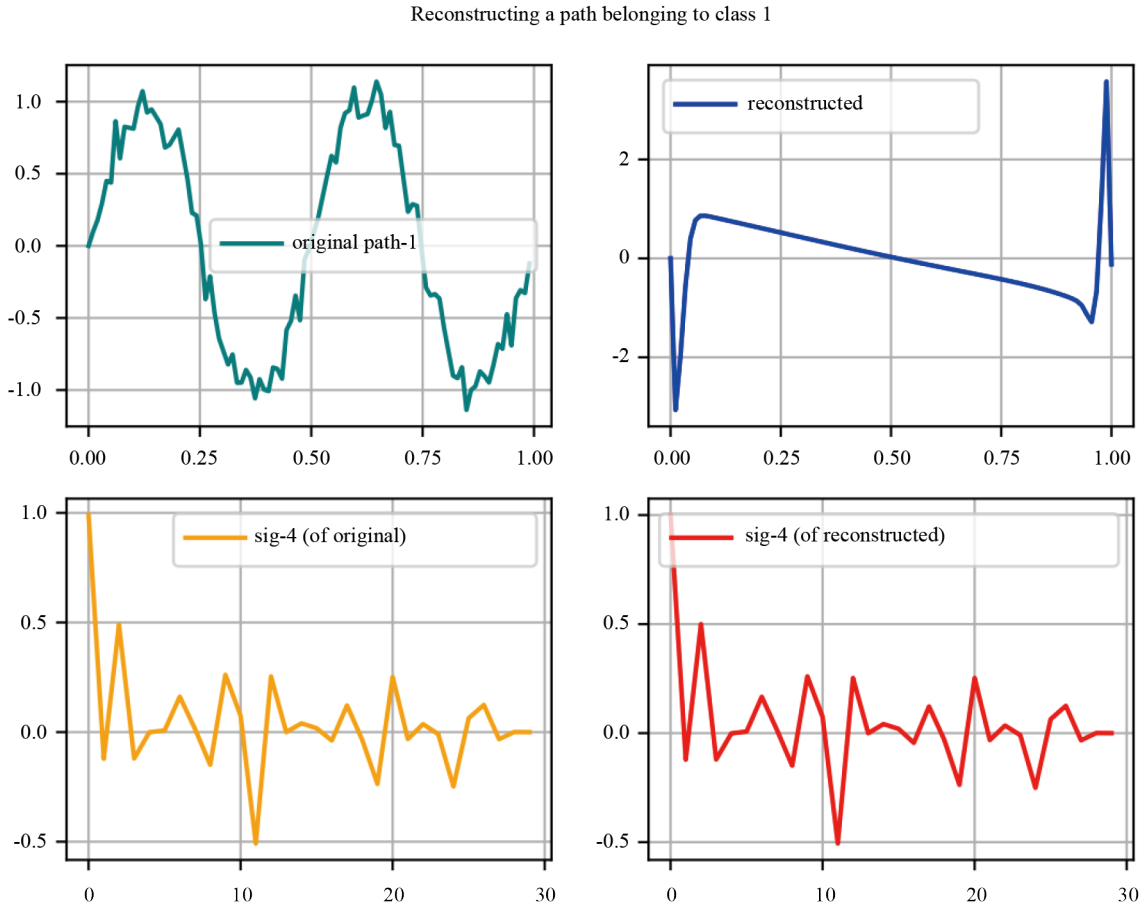

Figure 6. Reconstruction attempt for a noised double sine wave

In all cases, the relative error between the norm of the original signature (orange) and the norm of the signature of the reconstructed path (red) is below 1%, therefore the paths can actually be considered to have approximately the

same 4-signatures. Generally speaking, all the simulations produced "artificial spikes" around the extremes of the curves, hindering a faithful recovery of the information. It was also not possible to recover the "double frequencies" for the corresponding case, and both the pure sine and the noised sine shared very similar signatures. This is partially to be expected and understandable since this transform is actually well-connected to the "areas" covered by the curves.

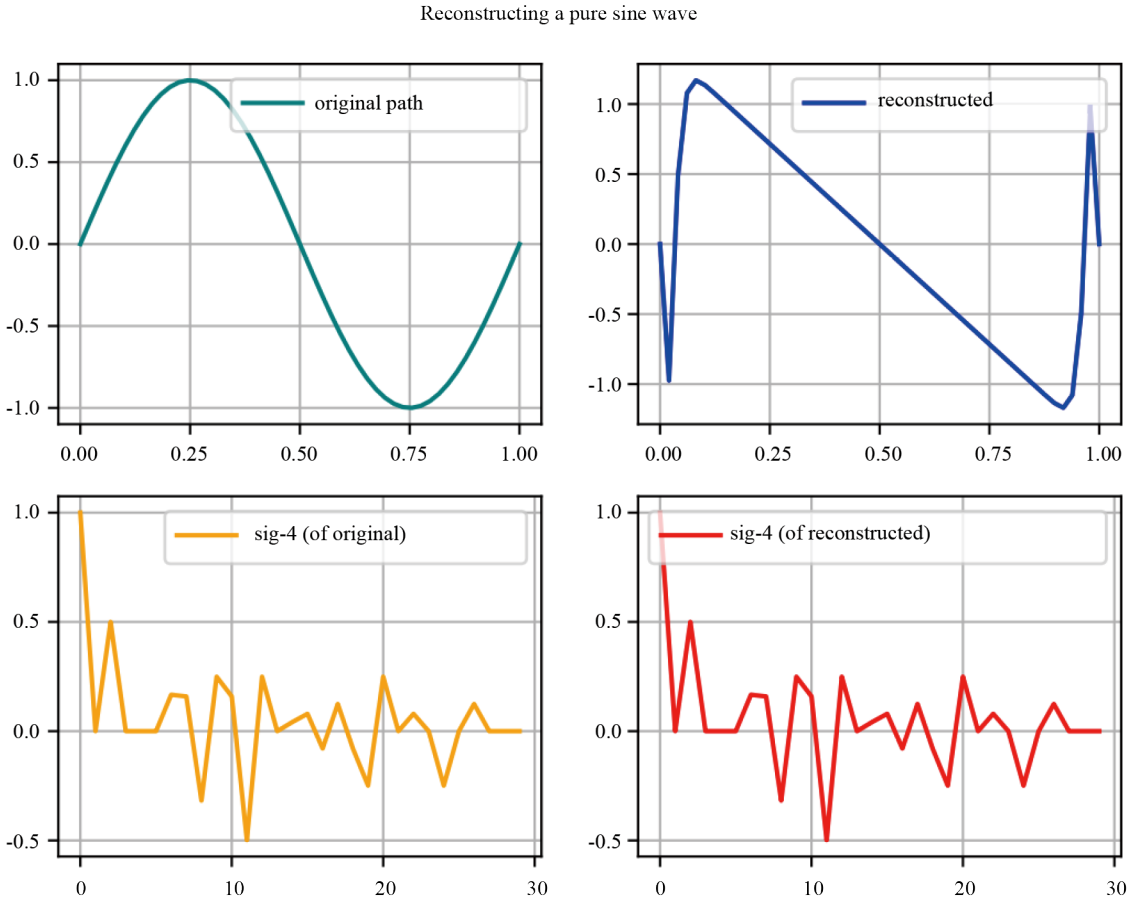

Figure 7. Reconstruction attempt for a single sine wave without noise

This is where Figure 8 comes into play. The layout is exactly the same as the previous one, but this time we considered the signature up to depth 6. This allowed for a much more faithful reconstruction but introduced an important problem: it was necessary to compute a total of $2^6 - 2 = 126$ coefficients, which is more than the points (around 80) composing the original signal! We are confident that despite these inconveniences, these techniques can really help in providing insights about the *interpretation* of the signature transform, and that it may eventually be possible to develop interesting solutions. We leave a more accurate analysis for possible future work.

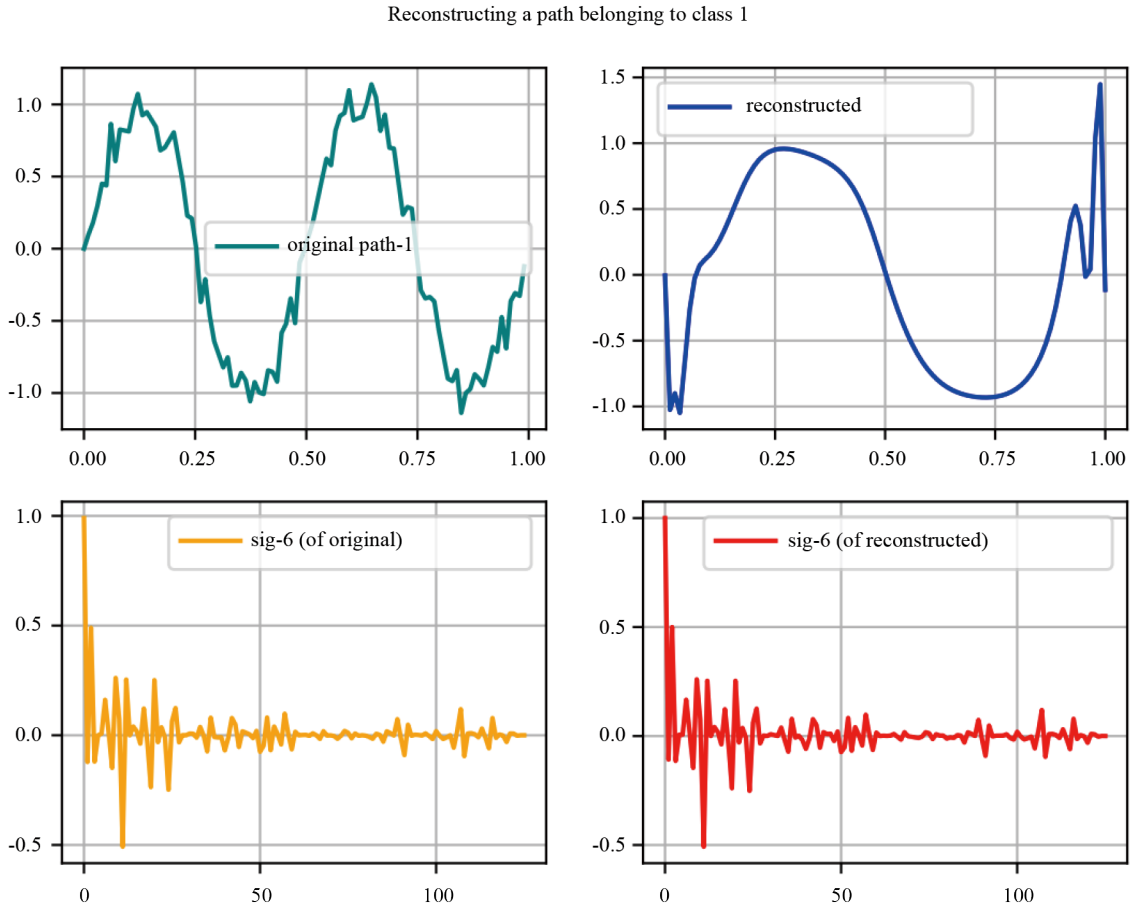

Figure 8. Deeper reconstruction attempt for double sine wave

7. Conclusions

The signature transform is a technique that can be considered hard to access due to its higher mathematical requirements. We attempted to offer a gradual and comprehensive introduction oriented towards machine learning and its applications, thanks to the choice of minimal and simple examples. In all of them, details are provided to allow reproducibility, using, for instance, the Python programming language with publicly available libraries, possibly even on computing machines with limited power.

When performing the transformation, a possible use case is for problems of time series classification. Preprocessing in this way does not require the data to have the same number of points and allows the use of simple linear classifiers that are easy to tune and computationally convenient. No particular *ad-hoc* knowledge (feature engineering) is required. The use of the multidimensional scaling algorithm is highlighted as a consistent method to isometrically visualize the database in two dimensions.

When reversing the transform, we mentioned a possible application in the context of signal reconstruction. Despite not obtaining the same promising results as in the classification case, our analysis might contribute to improving the overall intuitive understanding of what information is preserved when performing the signature transform.

More recent and advanced experiments are available in the author's openly available thesis [12]: for instance, it is possible to use the signature methodology combined with diffusion models that are typical in stochastic finance. Using this transform to classify agents' behavior in Reinforcement Learning has also been successful, as recently presented in [13].

Further progresses are in development and scheduled for upcoming works, involving topics like statistical analysis of real data and prediction strategies in energy financial markets.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Friz PK, Victoir NB. *Multidimensional Stochastic Processes as Rough Paths*. Cambridge: Cambridge University Press; 2010. Available from: https://doi.org/10.1017/CBO9780511845079.

[2] Lyons TJ, Caruana M, Lévy T. *Differential Equations Driven by Rough Paths*. Berlin, Heidelberg: Springer; 2007. Available from: https://doi.org/10.1007/978-3-540-71285-5.

[3] Chevyrev I, Kormilitzin A. A primer on the signature method in machine learning. *arXiv:160303788*. 2016. Available from: https://doi.org/10.48550/arXiv.1603.03788.

[4] Lyons T, McLeod AD. Signature methods in machine learning. *arXiv:220614674*. 2024. Available from: https://doi.org/10.48550/arXiv.2206.14674.

[5] Liao S, Ni H, Szpruch L, Wiese M, Sabate-Vidales M, Xiao B. Conditional sig-wasserstein GANs for time series generation. *arXiv:200605421*. 2023. Available from: https://doi.org/10.48550/arXiv.2006.05421.

[6] Wu B. Pattern recognition and classification in time series analysis. *Applied Mathematics and Computation*. 1994; 62(1): 29-45. Available from: https://doi.org/10.1016/0096-3003(94)90131-7.

[7] Cox MAA, Cox TF. Multidimensional scaling. In: *Handbook of Data Visualization*. Berlin, Heidelberg: Springer; 2008. p.315-347. Available from: https://doi.org/10.1007/978-3-540-33037-0_14.

[8] Fermanian A. Functional linear regression with truncated signatures. *arXiv:200608442*. 2022. Available from: https://doi.org/10.48550/arXiv.2006.08442.

[9] Kidger P, Lyons T. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. *arXiv:200100706*. 2020. Available from: https://doi.org/10.48550/arXiv.2001.00706.

[10] Lyons T, Xu W. Inverting the signature of a path. *arXiv:14067833*. 2018. Available from: https://doi.org/10.48550/arXiv.1406.7833.

[11] Fermanian A, Chang J, Lyons T, Biau G. The insertion method to invert the signature of a path. *arXiv:230401862*. 2023. Available from: https://doi.org/10.48550/arXiv.2304.01862.

[12] Paparella B. *The Signature Transform in Numerics and Machine Learning*. Ph.D. Thesis. Rheinische Friedrich-Wilhelms-Universität Bonn; 2024.

[13] Feiden A, Paparella B, Garcke J. Generalizing diversity with the signature transform. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. New York, NY, USA: Association for Computing Machinery; 2024. p.275-278. Available from: https://doi.org/10.1145/3638530.3654295.