An Elementary Introduction to Signature Methods and Rough Path Theory for ML

Thomas Li

April 2025

Abstract

How can Signature-based Methods be used in Machine Learning and what Modern Applications currently exist?

In many modern applications, data arrives as *streamed* sequences of observations, often indexed by time. Machine Learning seeks to systematically understand the behavior of this streamed data, despite the fact that it is often extremely irregular, nonstationary, and high-dimensional. **Signature Methods** and the broader framework of **Rough Path Theory** serve as a unique way to represent high dimensional data, yet preserve information about the data completely and uniquely (analogous to that of power series). It extract a hierarchy of coordinate-invariant, mathematically complete features via *iterated integrals*. These tools have become an extremely useful framework in learning from streams from a modern standpoint in data science.

This poster presents an *overview* of modern techniques in signature-based machine learning: including the **log-signature** and **expected signature** and how it may be applied to tasks like *time series classification* and regression. We additionally hope to present this mathematical framework in an intuitive, elementary fashion to improve comprehension.

Introduction of the Signature

We will begin by exploring the Picard Method of solving Differential Equations: Given a differential equation

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0, \text{we rewrite as } y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))\, dt,$$

and generating successive approximations:

$$y_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, y_n(t))\, dt,$$

starting from an initial guess $y_0(x) = y_0$.

Consider

$$\frac{dy}{dx} = y, y(0) = 1$$

Then:

$$\begin{aligned} y_0(x) &= 1 \\ y_1(x) &= 1 + \int_0^x 1\, dt = 1 + x \\ &\vdots \\ y_k(x) &= \sum_{n=0}^{k} \frac{x^n}{n!} \end{aligned}$$

taking k to infinity gives e^x.

Now consider a path $X : [a, b] \to \mathbb{R}^d$. Let $L(\mathbb{R}^d, \mathbb{R}^e)$ denote the vector space of linear maps from $\mathbb{R}^d$ to $\mathbb{R}^e$ (e by d matrices). Let $F : \mathbb{R}^e \to L(\mathbb{R}^d, \mathbb{R}^e)$ and $Y : [a, b] \to \mathbb{R}^e$
We will mainly examine the following Controlled Differential Equation:

$$dY_t = F(Y_t)dX_t$$

Taking the integral from a to t of both sides gives equivalently:

$$Y_t = Y_a + \int_a^t F(Y_s)dX_s$$

Now, $F = [F_1 \mid F_2 \mid \cdots \mid F_d]$ with each column vector $\in L(\mathbb{R}^e, \mathbb{R}^e)$. Similarly, $dX_s = \begin{bmatrix} dX_s^1 \\ \vdots \\ dX_s^d \end{bmatrix}$. Then we may write $Y_t = Y_a + \sum_{i=1}^{d} \int_a^t F_i(Y_s)dX_s^i$.

Performing Picard iterations now:

$$
\begin{aligned}
Y_t^0 &= Y_a, \\
Y_t^1 &= Y_a + \sum_{i=1}^{d} \int_a^t F_i(Y_s^0)\, dX_s^i = Y_a + \sum_{i=1}^{d} F_i(Y_a) \int_a^t dX_s^i, \\
Y_t^2 &= Y_a + \sum_{i=1}^{d} \int_a^t F_i(Y_s^1)\, dX_s^i \\
&= Y_a + \sum_{i=1}^{d} F_i(Y_a) \int_a^t dX_s^i + \sum_{i,j=1}^{d} F_i(F_j(Y_a)) \int_a^t \int_a^s dX_u^j\, dX_s^i, \\
Y_t^n &= Y_a + \sum_{i=1}^{d} \int_a^t F_i(Y_s^{n-1})\, dX_s^i \\
&= Y_a + \sum_{k=1}^{n} \sum_{i_1,\dots,i_k=1}^{d} F_{i_k}(\dots(F_{i_1}(Y_a))\dots) \int_{a<t_1<\cdots<t_k<t} dX_{t_1}^{i_1} \dots dX_{t_k}^{i_k},
\end{aligned}
$$

Taking $n \to \infty$:

$$
Y_t = Y_a + \sum_{k=1}^{\infty} \sum_{i_1,\dots,i_k=1}^{d} F_{i_k}(\dots(F_{i_1}(Y_a))\dots) \int_{a<t_1<\cdots<t_k<t} dX_{t_1}^{i_1} \dots dX_{t_k}^{i_k}.
$$

This expansion naturally suggests the concept of the **Signature** of a path.

Observe that the terms

$$
\int_{a<t_1<\cdots<t_k<t} dX_{t_1}^{i_1} \cdots dX_{t_k}^{i_k}
$$

depend only on the path X and not on the particular dynamics F or initial condition Y_a. They capture the iterated interactions along the path and encode the sequential structure of its evolution. Collecting all such iterated integrals across all $k \geq 0$ gives the **Signature** of the path X over $[a, t]$, denoted by $S(X)_{a,t}$.

Formally, let $X = (X^1, \dots, X^d) : [a, t] \to \mathbb{R}^d$ be a continuous d-dimensional path. We define the **Signature** of X over $[a, t]$ as the sequence of iterated integrals:

$$
\begin{aligned}
S(X)_{a,t} = \Bigg(1, \int_a^t dX_s, \int_{a<s_1<s_2<t} dX_{s_1} \otimes dX_{s_2}, \\
\int_{a<s_1<s_2<s_3<t} dX_{s_1} \otimes dX_{s_2} \otimes dX_{s_3}, \dots \Bigg) \in T(\mathbb{R}^d).
\end{aligned}
$$

Here, $dX_s = (dX_s^1, \dots, dX_s^d)$ is a vector of differentials, and the $\otimes$ symbol denotes the tensor product. At level k, the signature contains one entry for every multi-index $(i_1, \dots, i_k) \in \{1, \dots, d\}^k$, corresponding to the iterated integral

$$S(X)_{a,t}^{i_1,\ldots,i_k} := \int_{a<s_1<\cdots<s_k<t} dX_{s_1}^{i_1} \cdots dX_{s_k}^{i_k}.$$

Thus, the second level, for instance, contains terms like:

$$\int_{a<s_1<s_2<t} dX_{s_1}^i dX_{s_2}^j \quad \text{for all } 1 \leq i, j \leq d,$$

which together form a $d \times d$ matrix (a 2-tensor).

More concretely, we can write:

$$\begin{aligned} S(X)_{a,t} = \Big(1, \int_a^t dX_s^1, \ldots, \int_a^t dX_s^d, \\ \int_{a<s_1<s_2<t} dX_{s_1}^1 dX_{s_2}^1, \int_{a<s_1<s_2<t} dX_{s_1}^1 dX_{s_2}^2, \\ \ldots, \int_{a<s_1<s_2<t} dX_{s_1}^d dX_{s_2}^d, \ldots \Big). \end{aligned}$$

where each entry corresponds to a specific ordered combination of components and integration times.

Just as truncating terms in the ODE example provides a good approximation to e^x, we can similarly truncate the **Signature terms** to obtain an effective approximation of the path's behavior.

Significant Properties

Theorem (Uniqueness of Signatures up to Tree-Like Equivalence):
Let $X, Y : [0, T] \to \mathbb{R}^d$ be two continuous paths of bounded variation or rough paths. $S(X) = S(Y)$ if and only if X and Y are tree-like equivalent. (Hambly-Lyons, 2010)
For the more advanced reader: For $p \geq 1$, the p-variation of a path $X : [0, T] \to \mathbb{R}^d$ is:

$$\|X\|_{p-\text{var};[0,T]} = \sup_P \left(\sum_{[t_i, t_{i+1}] \in P} \|X_{t_{i+1}} - X_{t_i}\|^p \right)^{1/p}.$$

Chen was the first to look at these iterated integrals as Group-like structures but not unique (1957). Hambly-Lyons proved that when the p-variation is finite, the uniqueness theorem applies (2010). Bojnik-Nakamura extended the uniqueness theorem to $p \geq 1$ (all paths) under specific conditions (2023).
Ultimately, this shows that the Signatures provide a Universal Feature Set for all paths. Given enough signature terms, we can approximate any reasonable function on paths

Factorial Decay of High Order Terms
The projection of the kth-level signature from the tensor algebra decays factorially:

$$\|S_k(X)_{a,t}\| \leq \frac{(\|X\|_{1-\text{var};[a,t]})^k}{k!}$$

(Lyons, 2007). This justifies that truncation of the signature up to a relatively high level will capture sufficient information about the path. Thus, when doing regression tasks or Machine Learning, we do not lose crucial information when truncating our signature terms.

Parameterization Invariance
Given a path $X : [a, b] \to \mathbb{R}^d$, if we take $\phi : [c, d] \to [a, b]$, then $X_{\phi(t)}$ on $t \in [c, d]$ is a reparameterization of the path. It is true that $S(X_t)_{a,b} = S(X_{\phi(t)})_{c,d}$. This is a trivial proof: Set $u = \phi(t)$ on the reparameterized signature and Change of Variables on the integrals shows equivalence.

Chen's Identity
Let $a < b < c$ and $X : [a, c] \to \mathbb{R}^d$: the term

$$S(X)^{i_1,\ldots,i_k}_{a,c} = \sum_{m=0}^{k} S(X)^{i_1,\ldots,i_m}_{a,b} S(X)^{i_{m+1},\ldots,i_k}_{b,c}$$

Example: $S(X)^{1,2}_{0,2} = S(X)^{1,2}_{1,2} + S(X)^{1}_{0,1} S(X)^{2}_{1,2} + S(X)^{1,2}_{0,1}$

Different Methods

Derivation of the Log-Signature

Formal power series in the tensor algebra Let V be a finite-dimensional vector space with basis $\{e_1, \ldots, e_d\}$. An arbitrary *formal power series*

$$y \;=\; \sum_{k=0}^{\infty} \sum_{i_1,\ldots,i_k=1}^{d} \lambda_{i_1,\ldots,i_k}\; e_{i_1} \otimes \cdots \otimes e_{i_k} \;\in\; T((V))$$

is specified by its *coefficients* $\lambda_{i_1,\ldots,i_k} \in \mathbb{R}$

$$e_{i_1} \otimes \cdots \otimes e_{i_k} \;\in\; V^{\otimes k}$$

is the k-fold tensor product of the basis vectors. To make sense of this for readers not acquainted with the Tensor Algebra:

$k = 0 : 1$ by convention

$k = 1 :$ A linear combination of basis vectors e_i with real coefficients λ_i

$k = 2 :$ Ordered pairs (e_i, e_j) that specify the entries of a matrix whose values are $\lambda_{i,j}$

$\vdots$

In essence, the tensor product specifies the location of the entry in the k-dimensional tensor and lambda encodes the value at that location.

Consider $\frac{1}{1-z} = \sum_{n\geq 0} z^n$. Then integrating and substituting gives

$$\ln(z) = \sum_{n\geq 1} (-1)^{n-1} \frac{(z-1)^n}{n}$$

Formally, this holds for any algebraic object z. When it is an object of the tensor algebra, we must have:

$$\ln(z) = \sum_{n\geq 1} (-1)^{n-1} \frac{(z-1)^{\otimes n}}{n}$$

This should make sense. The tensor product takes two entries from $e_{i_1} \otimes \cdots \otimes e_{i_k}$ and $e_{j_1} \otimes \cdots \otimes e_{j_{k'}}$ and concatenates them to specify a location of a higher dimensional tensor: $e_{i_1} \otimes \cdots \otimes e_{i_k} \otimes e_{j_1} \otimes \cdots \otimes e_{j_{k'}}$. It's entry (coefficient) is the product of the coefficients of the two smaller tensors. Proving rigorously that tensor product is the formal analogy is beyond the scope, however.

Now the *signature* of a path $X : [a, b] \to \mathbb{R}^d$,

$$\mathbf{S}(X)_{a,b} = \sum_{k=0}^{\infty} \sum_{i_1,\ldots,i_k=1}^{d} \left(\int_{a<s_1<\cdots<s_k<b} dX_{s_1}^{i_1} \cdots dX_{s_k}^{i_k} \; e_{i_1} \otimes \cdots \otimes e_{i_k} \right).$$

Its *log-signature* is defined by

$$\log \mathbf{S}(X)_{a,b} \;=\; \sum_{n=1}^{\infty} (-1)^{\,n-1} \, \frac{\big(\mathbf{S}(X)_{a,b} - 1\big)^{\otimes n}}{n} \;=\; \frac{(\mathbf{S}(X)_{a,b} - 1)^{\otimes 1}}{1} - \frac{(\mathbf{S}(X)_{a,b} - 1)^{\otimes 2}}{2} + \ldots$$

$$= \sum_{i=1}^{d} S^i(X)_{a,b} \, e_i \;+\; \sum_{i,k=1}^{d} (S^{i,k}(X) - \frac{1}{2} S^i(X)_{a,b} S^k(X)_{a,b})(e_i \otimes e_k) + \ldots$$

This convention is intuitive to store as a vector. **However, the terms are often reorganized using lie brackets:** $[e_i, e_j] = e_i \otimes e_j - e_j \otimes e_i$**.** This expansion is not a trivial problem either (related to the Baker-Campbell-Hausdorff

formula). We will show a completely elementary derivation of the second level of the log-signature:

$$\sum_{i,k=1}^{d} (S^{i,k}(X) - \frac{1}{2} S^i(X)_{a,b} S^k(X)_{a,b})(e_i \otimes e_j)$$

$$S^i(X)S^k(X) = \int_{a \le t_1, t_2 \le b} dX^i_{t_1} dX^k_{t_2} = \int_{[a \le t_1 < t_2 \le b] + [a \le t_2 < t_1 \le b] + [t_1 = t_2]} dX^i_{t_1} dX^k_{t_2}$$

The square region on $[a, b]^2$ is split in 3 regions: upper half, lower half, and diagonal. The diagonal contributes 0 (no area).

$= S^{i,k}(X) + S^{k,i}(X)$

Thus: $S^{i,k} - \frac{1}{2} S^i S^k = \frac{1}{2} \left(S^{i,k} - S^{k,i} \right).$

$$\sum_{i,k=1}^{d} \frac{1}{2} \left(S^{i,k} - S^{k,i} \right) (e_i \otimes e_k) = \sum_{i,k=1}^{d} \frac{1}{2} \left(S^{i,k} - S^{k,i} \right) \left(\frac{1}{2} \underbrace{(e_i \otimes e_k + e_k \otimes e_i)}_{\text{symmetric}} + \frac{1}{2} \underbrace{(e_i \otimes e_k - e_k \otimes e_i)}_{[e_i, e_k]} \right).$$

The symmetric term goes to 0 as entries at each $e_1 \otimes e_2$ cancel. We are left with:

$$\begin{aligned}
&\sum_{i,k=1}^{d} \frac{1}{4} \left(S^{i,k} - S^{k,i} \right) [e_i, e_k] = \sum_{1 \le i < k \le d} \frac{1}{4} \left(S^{i,k} - S^{k,i} \right) [e_i, e_k] + \\
&\sum_{1 \le k < i \le d} \frac{1}{4} \left(S^{i,k} - S^{k,i} \right) [e_i, e_k] + \sum_{1 \le i \le d} \frac{1}{4} \left(S^{i,i} - S^{i,i} \right) [e_i, e_i] \\
&= \sum_{1 \le i < k \le d} \frac{1}{4} \left(S^{i,k} - S^{k,i} \right) [e_i, e_k] + \sum_{1 \le i < k \le d} \frac{1}{4} \left(S^{k,i} - S^{i,k} \right) (-[e_i, e_k]) \\
&= \frac{1}{2} \sum_{1 \le i < k \le d} \left(S^{i,k} - S^{k,i} \right) [e_i, e_k].
\end{aligned}$$

The logarithm of the signature simplifies to:

$$\log \mathbf{S}(X)_{a,b} = \sum_{i=1}^{d} S^i e_i + \frac{1}{2} \sum_{i<k} \left(S^{i,k} - S^{k,i} \right) [e_i, e_k] + \cdots.$$

In practice, we will save the entries of the coefficients of the lie algebra object in our array or vector! The motivation of the log signature is that, oftentimes in the Signature, certain terms are not linearly dependent. We saw that in $S^i S^k = S^{ik} + S^{ki}$

Expected Signature

- Many real-world datasets are generated by an **underlying stochastic process** (e.g., stock prices). We have access to data, which is produced

by this complex process. Hence, if we can understand the underlying stochastic process, we can make predictions!

- Given N sample paths $\{X_i\}_{i=1}^N$ on the same interval of time, we can compute each path's signature $S(X_i)$. Then, the expected signature is

$$\mathbb{E}\big[S(X)\big] \;=\; \frac{1}{N}\sum_{i=1}^{N} S(X_i),$$

  which characterizes the average behavior of the paths and captures information about the distribution of the underlying process.

Suppose we have a time interval $J = [0, T]$ and $X : J \to \mathbb{R}^d$ is a path-valued input, while $Y : J \to \mathbb{R}^m$ is an output path. We observe N input-output pairs (X_i, Y_i) over J, and assume a regression model (given an input path, I can predict an output path):

$$Y_i = f(X_i) + \epsilon_i, \quad \text{for } i = 1, \ldots, N,$$

where f is an unknown function and ϵ_i is noise. The expected value of Y, given an input X is with $\mathbb{E}[Y_i \mid X_i] = f(X_i)$, which is what we want to predict.

The **expected signature method** replaces paths with their signatures:

$$\mathbb{E}[S(Y_i) \mid S(X_i)] \approx L(S(X_i)),$$

or that

$$S(Y_i) = L(S(X_i)) + \epsilon$$

where $L : T((\mathbb{R}^d)) \to T((\mathbb{R}^m))$ is a linear map.

Linear regression setup:

$$\begin{aligned} \text{Given:} \quad & \{(S(X_i), S(Y_i))\}_{i=1}^N \\ \text{Learn:} \quad & L : T((\mathbb{R}^d)) \to T((\mathbb{R}^m)) \\ \text{Fit:} \quad & S(Y_i) \approx L(S(X_i)) \quad \text{for each } i = 1, \ldots, N. \end{aligned}$$

Once L is learned (e.g., using least squares or regularized regression), for a new input path X_{new}, we can predict the output signature:

$$\hat{S}(Y_{\text{new}}) = L(S(X_{\text{new}})).$$

From this predicted signature, one may attempt to reconstruct Y_{new}. Oftentimes, it is useful to set the output path, to be equal to the future of the input path.

Applications

Brownian Motion

A fundamental stochastic process where increments follow independent normal distributions. For our study, we generate discrete paths:

$$X_t = \sum_{i=1}^{t} \epsilon_i, \ \epsilon_i \sim \mathcal{N}(0, \Delta^2) \tag{1}$$

with two volatility scales $\Delta \in \{0.10, 0.12\}$.

Classification Challenge

Distinguishing between these processes requires detecting subtle differences in path volatility. Traditional approaches using raw path data often struggle with temporal dependency structure.

1 Methodology

1.1 Data Generation

- Generated 1,000 paths (500 per class) with $T = 70$ time steps
- Stratified split: 70% training, 30% testing
- **Raw Features**: We collect the raw time-series values into a matrix

$$\mathbf{X}_{\text{raw}} = \begin{bmatrix} X_1(t_1) & X_1(t_2) & \cdots & X_1(t_T) \\ X_2(t_1) & X_2(t_2) & \cdots & X_2(t_T) \\ \vdots & \vdots & \ddots & \vdots \\ X_N(t_1) & X_N(t_2) & \cdots & X_N(t_T) \end{bmatrix} \in \mathbb{R}^{N \times T},$$

  where $X_i(t_j)$ is the j-th observation of the i-th sample path.

- **Signature Features**:
  1. **Lead–Lag Transformation**: Each 1D series $X_i(t)$ is lifted to a 2D path
  $$\widetilde{X}_i = \left\{ \big(X_i(t_j),\, X_i(t_{j-1})\big) \right\}_{j=1}^{T} \in \mathbb{R}^{T \times 2}.$$
  2. **Log-Signature Computation**: From each 2D path $\widetilde{X}_i$ we compute its truncated log-signature
  $$\log S(\widetilde{X}_i)_{0,T} \in \mathbb{R}^D,$$
  up to order 5. We then assemble these into the signature feature matrix. Each $\log S(\widetilde{X}_i)_{0,T}$ is a horizontal vector of signature terms

up to the fifth level (go back to the definition of the log-sig...you will see the 1st and 2nd levels and the entries in there).

$$\mathbf{X}_{\text{sig}} = \begin{bmatrix} \log S(\widetilde{X}_1)_{0,T} \\ \log S(\widetilde{X}_2)_{0,T} \\ \vdots \\ \log S(\widetilde{X}_N)_{0,T} \end{bmatrix} \in \mathbb{R}^{N \times D}.$$

Model Configuration and Evaluation Metrics

Random Forest "Recipe" (identical for both raw- and signature-based features)

- **200 trees:** Think of this as consulting 200 "mini-experts" (decision trees). Each tree votes on the class, and the forest takes the majority vote. More trees generally yield a more stable, reliable decision.
- **Maximum depth = 15:** Each tree can ask up to 15 successive yes/no questions about the each set of features for each path (e.g. "Is feature 37 > 0.42?"). Limiting depth prevents any single tree from memorizing every tiny fluctuation ("overfitting") $\rightarrow$ learns overall patterns
- **Feature sampling = $\sqrt{\text{total features}}$:** At each split, each tree only looks at a random subset of $\sqrt{p}$ features (if there are p total). This injects randomness so that each tree sees a different "view" of the data, reducing correlated errors.

Performance Metrics

- **Accuracy**

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

  E.g. an accuracy of 0.90 means the model's vote matches the true class 90% of the time.

- **Precision (for "positive" class)**

$$\text{Precision} = \frac{\text{True positives}}{\text{True positives} + \text{False positives}}.$$

  Of all examples labeled "1" by the model, what fraction are truly class 1? High precision means "when we predict 1, we're usually right."

- **Recall (Sensitivity)**

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negatives}}.$$

  Of all actual class 1 examples, what fraction did the model correctly identify? High recall means "we catch most of the 1's."

Using all three metrics together guards against misleading conclusions:

- A trivial model might achieve high accuracy by always predicting the majority class, yet have zero recall for the minority class.
- A model that rarely predicts "1" could have high precision (it's only guessing 1 when sure) but low recall (it misses most real positives).

2 Results

2.1 Classification Performance

Table 1: Comparative Classification Results

Feature Type	Accuracy	Precision	Recall	F1-score
Raw Path Data	57.33%	0.56	0.68	0.61
Signature Features	85.0%	0.89	0.80	0.84

If we now repeat the experiment 5 times:

Metric	Mean Accuracy	Standard Deviation
Raw Data Accuracy	58.07%	$\pm 2.81\%$
Signature Accuracy	84.93%	$\pm 1.18\%$
Absolute Improvement	26.87%	
Relative Improvement	46.27%	

Table 2: Comparison of classification performance over 5 runs.

Our full algorithm is available as an executable Colab notebook at this link.

3 Conclusion

Path signatures offer superior classification performance for distinguishing stochastic processes with subtle parameter differences. The Brownian Motion is a classic example of quite a chaotic set of data, and path signature methods are able to perform significantly better. In fact, signatures have even been used in the following applications:

- Financial time series analysis: anomaly detection... even detecting market manipulation!
- Brain Computer Interfaces: EEG signals are analyzed using signatures for better performance for the disabled

- Chinese Character Handwriting Recognition: Signatures have also been used to recognize Chinese handwriting due to their unique representation of handwriting paths!

References

1. Ilya Chevyrev and Andrey Kormilitzin, "A Primer on the Signature Method in Machine Learning," arXiv:1603.03788, 2016.

2. Terry Lyons and David J. C. MacKey, "The Rough Paths Theory and Its Applications," arXiv:2206.14674, 2022.

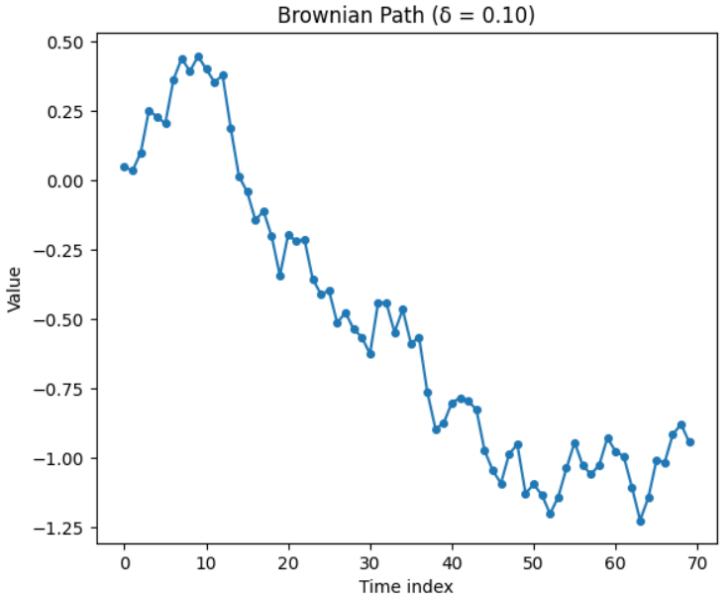
3. Victoria Galindo Ana, "Master's Thesis: Signature Methods for Time Series Analysis," University of Barcelona, 2023. `https://diposit.ub.edu/dspace/bitstream/2445/215450/2/tfm_victoria_galindo_ana.pdf`

4. Terry Lyons and Nicolas Victoir, "Cubature on Wiener Space," arXiv:1309.0260, 2013.

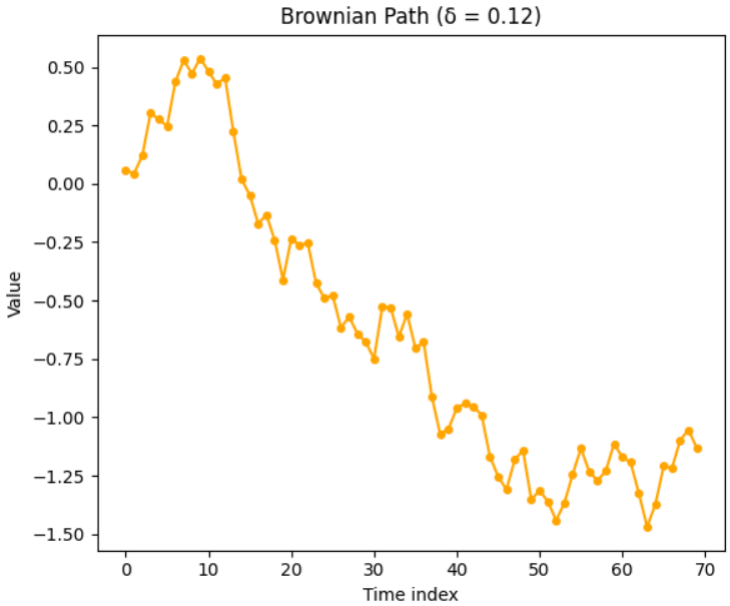
5. B. Graham, "Sparse Arrays of Signatures for Online Character Recognition," arXiv:1308.0371, 2013.

6. "Time Series Forecasting and Simulations in Python: Signature Transformation Method," OpenDataScience. `https://opendatascience.com/time-series-forecasting-and-simulations-python-signature-transformation-method/`

7. Pavel Zapolskii, "From Data Science and ML Specialist to Senior Trading Researcher," Exness Research Blog, 2024.

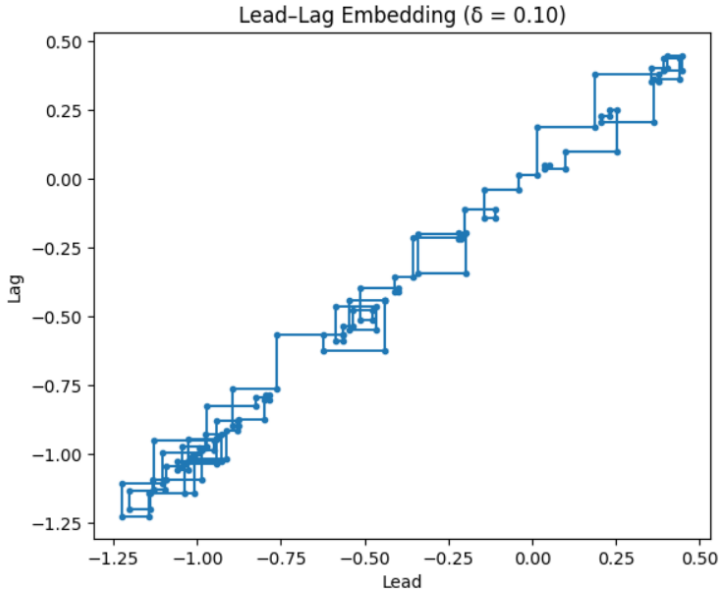


(a) Brownian Motion Path with $\delta = 0.10$

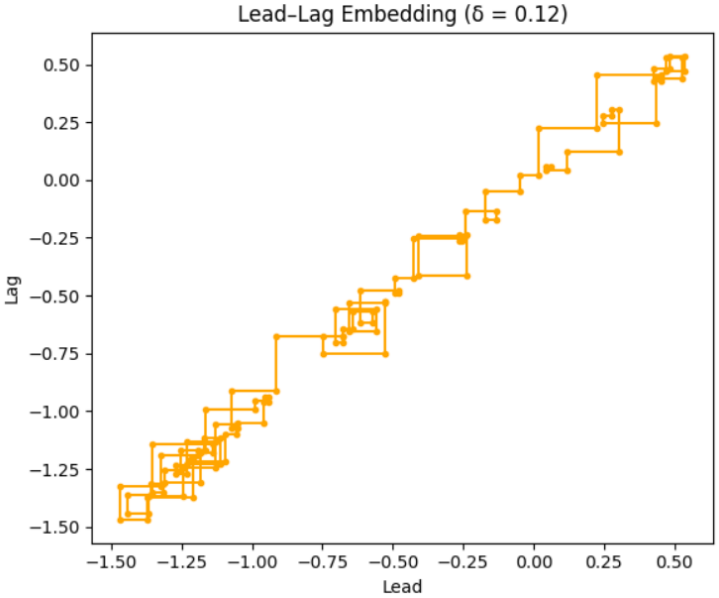


(b) Brownian Motion Path with $\delta = 0.12$

Figure 1: Visual comparison of two Brownian motion paths with differing step sizes ($\delta = 0.10$ and $\delta = 0.12$) and their respective 2D lead-lag embeddings.



(a) Lead Lag Transformation of Path with $\delta = 0.10$



(b) Lead Lag Transformation of Path with $\delta = 0.12$

Figure 2: Visual comparison of Lead Lag Paths with differing step sizes ($\delta = 0.10$ and $\delta = 0.12$) and their respective 2D lead-lag embeddings.