UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Application of the Signature Method in Time Series and Financial Data Streams

---

Author:
Ana VICTORIA GALINDO

Supervisor:
Dr. Josep VIVES SANTA EULALIA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 28, 2024

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

Master in Fundamental Principles of Data Science

Application of the Signature Method in Time Series and Financial Data Streams

by Ana VICTORIA GALINDO

This thesis focuses into the signature method's role as a robust tool in data science, specifically within the realms of time series analysis and financial data streams. Originating from rough paths theory, the signature method offers a comprehensive representation of sequential data, effectively capturing intricate patterns and dependencies crucial for advanced modeling and predictive analytics.

Establishing a solid theoretical foundation, this thesis explores how the signature method transforms raw time series data into structured representations that preserve essential dynamic information. Through theoretical insights and practical illustrations, the thesis demonstrates the method's efficacy in enhancing model classification, temporal segmentation, and understanding complex model structures.

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. Josep Vives, for his guidance and for proposing such an interesting research topic. His support and advice have been invaluable throughout this process.

I also want to thank my family—Laura, Victor, Eva, Ingrid, Fernando and Benito—for always supporting me and believing in me. Their unwavering confidence in my abilities has been a constant source of motivation.

Finally, I would like to extend my deepest thanks to my classmate from the mathematics degree, Oscar Romero. For his final degree project, he researched a very similar topic and has greatly assisted me in resolving issues and discussing various aspects of my work. His collaboration and insights have been very helpful.

Contents

Chapter 1

Introduction

This thesis explores the utilization of the signature method as a powerful tool in data science, specifically in the realm of time series analysis and financial data streams. This thesis aims to establish a theoretical foundation for the signature method and demonstrate its practical applications through three distinct studies.

The signature method offers a unique approach to capturing complex patterns and dependencies within sequential data. Originally developed in the field of rough paths theory, the signature method provides a concise yet comprehensive representation of data streams, facilitating enhanced model classification and predictive analytics.

This thesis begins with an exploration of the theoretical base of the signature method, explaining its role in transforming raw time series data into structured representations that preserve essential information about the data's dynamics. Subsequently, it presents three applied studies to illustrate the method's efficacy:

- Model Classification: Comparing traditional Autoregressive (AR), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA) models using signature-based features.
- Temporal Segmentation: Classifying crude oil data streams based on 30-minute intervals using signature features, demonstrating the method's applicability in temporal segmentation tasks.
- Equivalence Demonstration: Investigating how AR models can be viewed as a special case within the broader framework of Expected Signature (ES) models, highlighting the method's versatility in capturing varied model structures.

Each study showcases the signature method's ability to extract complex temporal information into structured and interpretable features, thereby enhancing understanding and predictive capabilities across diverse domains. By bridging theoretical insights with practical applications, this thesis offers insights into leveraging signature-based approaches for complex data analysis. For access to the code and implementations used in these studies, please refer to my GitHub Repository.

Chapter 2

Time Series

According to the book *Introduction to Time Series and Forecasting*, "A time series model for the observed data x_t is a specification of the joint distributions (or possibly only the means and covariances) of a sequence of random variables X_t of which x_t is postulated to be a realization " (Brockwell and Davis, 2016). A process $\{X_j, j \in \mathbb{Z}\}$ is strictly stationary if for any $k_1, \ldots, k_n$ and l the vectors

$$(X_{k_1}, \ldots, X_{k_n})$$

and

$$(X_{k_1+l}, \ldots, X_{k_n+l})$$

have the same law. We say that a process is stationary if

- $\mathbb{E}[X_k] = \mu \in \mathbb{R}, \quad \forall k \in \mathbb{Z}$, and
- $\mathbb{C}(X_k, X_{k+l}) = \gamma(l), \quad \forall k, l \in \mathbb{Z}$, where γ is a function defined on $\mathbb{N}$

It is important to define what is a white noise. A process $\{X_k, k \geq 1\}$ is termed as white noise if all its variables have an expectation of μ, a variance of σ^2, and are uncorrelated. We say that the white noise is centered when $\mu = 0$.

2.1 Autoregressive models (AR)

Let $Z = \{Z_j, j \in \mathbb{Z}\}$ be a centered white noise with variance σ^2 and $\{Y_j, j \in \mathbb{Z}\}$ a time series. Define the operator B such as $BY_j := Y_{j-1}$. With this information we can now define the Autoregressive model AR(1).

Definition 2.1.1. *The AR(1) equation is*

$$Y_j = \phi B Y_j + Z_j, \quad \phi \in (-1, 1),$$

which is equivalent to $Y_j = \phi Y_{j-1} + Z_j$

We can generalize this expression to define the AR(p) model. Let $\Phi(x)$ be a polynomial of degree p. Note that we can always write Φ as

$$\Phi(x) = 1 - \phi_1 x - \phi_2 x^2 - \cdots - \phi_p x^p$$

just by dividing the polyonomial by the independent coefficient.

Recall that a polynomial $\Phi(x)$ is invertible if there exists a series $\sum_{i=0}^{\infty} \psi_i x^i$ such that

$$\Phi(x) \sum_{i=0}^{\infty} \psi_i x^i = 1$$

with $\sum_{i=0}^{\infty} \psi_i^2 < \infty$.

Now we can define an AR(p) model:

Definition 2.1.2. *Let Φ be an invertible polynomial of degree p. A process $\{Y_j, j \in \mathbb{Z}\}$ is an AR(p) model if it is stationary and satisfies*

$$\Phi(B)Y_j = Z_j, \quad j \in \mathbb{Z}, \quad Z_j.$$

This expression is equivalent to $Y_j - \phi_1 Y_{j-1} - \phi_2 Y_{j-2} - \cdots - \phi_p Y_{j-p} = Z_j$

2.2 Moving Average models (MA)

Moving Average models are based on the idea that the observed value of a time series variable at any given time point is a linear combination of the current and past random error terms. Unlike autoregressive (AR) models, which rely on past values of the variable itself, MA models use past values of the error terms to model the data.

Definition 2.2.1. *Let $Z = \{Z_j, j \in \mathbb{Z}\}$ be a white noise with variance σ^2 and $q \geq 0$ A process $\{Y_j, j \in Z\}$ is a MA(q) model if it has the following form:*

$$Y_j = Z_j + \theta_1 Z_{j-1} + \cdots + \theta_q Z_{j-q}$$

with $\theta_1, \ldots, \theta_q$ being real numbers.

2.3 AutoRegressive Moving Average model(ARMA)

An ARMA (AutoRegressive Moving Average) process is a commonly used model in time series analysis that combines the two models seen before: the AutoRegressive (AR) model and the Moving Average (MA) model.

Definition 2.3.1. *Let p and q be two natural numbers, let Z be a white noise with mean 0 and variance σ^2 and $\Phi_p(\cdot)$ and $\Theta_q(\cdot)$ two invertible polynomials without roots in common with degree p and q respectively. The process $\{Y_j, j \in \mathbb{Z}\}$ is an ARMA(p, q) id it satisfies the equation*

$$\Phi_p(B)Y_j = \Theta_q(B)Z_j, \quad j \in \mathbb{Z},$$

which is equivalent to the following expression:

$$Y_j - \phi_1 Y_{j-1} - \cdots - \phi_p Y_{j-p} = Z_j + \theta_1 Z_{j-1} + \cdots + \theta_q Z_{j-q}$$

2.4 AutoRegressive Integrated Moving Average model (ARIMA)

An ARIMA model is an extension of the ARMA model that is used to analyze and forecast time series data that may exhibit non-stationarity. The components of the ARIMA models are the following:

- AutoRegressive (AR) Component: This part involves regressing the variable on its own lagged (past) value as seen in section 2.1.
- Integrated (I) Component: This part involves differencing the series to make it stationary. Differencing the series d times is denoted as $I(d)$.
- Moving Average (MA) Component: As seen above in section 2.2, this part models the error term as a linear combination of past error terms.

Now we can define an ARIMA(p, d, q) model:

Definition 2.4.1. *Let X be a process $\{X_j, j \in \mathbb{Z}\}$ and p, d, q three fixed natural numbers. We say that X is an ARIMA(p, d, q) model if*

$$Y_j = (Id - B)^d X_j, \quad j \in \mathbb{Z}$$

is an ARMA(p, q) model.

Chapter 3

The Signature

3.1 The Signature

The signature method is a mathematical framework derived from rough path theory, designed to handle and analyze sequential data, such as time series. It transforms complex, high-dimensional data into a structured format that captures the essential patterns and behaviors of the sequence.

3.1.1 Previous Concepts

Before starting with the definition of signature we need to introduce some previous concepts.

Definition 3.1.1. *We define a **path** X in $\mathbb{R}^d$ as a continuous mapping from an interval $[a, b]$ to $\mathbb{R}^d$. The notation used will be $X_t = X(t)$, to highlight the importance on the dependence on the parameter $t \in [a, b]$.*

Definition 3.1.2. *A **smooth path** is a path which has derivatives of all orders.*

During this project we will refer to a paths assuming that they are always piecewise differentiable.

A simple example of a smooth path in $\mathbb{R}^2$ is the following and its representation can be seen in Figure 3.1:

$$X_t = \{X_t^1, X_t^2\} = \{\cos(t), 0.5\sin(t)\}, \quad t \in [0, 2\pi].$$

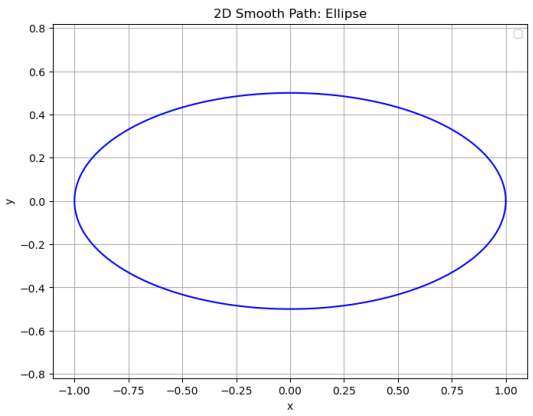


FIGURE 3.1: Two-dimensional smooth path

In $\mathbb{R}^d$ we can generalize the expression of the path to

$$X : [a,b] \to \mathbb{R}^d, \quad X_t = \{X_t^1, X_t^2, \dots, X_t^d\}.$$

With this understanding, we can now introduce the concept of the path integral.

Definition 3.1.3. *Let $X : [a,b] \to \mathbb{R}$ be a one-dimensional path and $f : \mathbb{R} \to \mathbb{R}$ be a function. We can define the **path integral of** X **against** f as*

$$\int_a^b f(X_t) dX_t = \int_a^b f(X_t) \dot{X}_t dt, \tag{3.1.1}$$

where $\dot{X}_t = X_t / dt$.

In the equation (3.1.1) $f(X_t)$ is also a path defined from $[a,b]$ to $\mathbb{R}$. In addition we can define the path integral of one real-valued path against another.

Definition 3.1.4. *Let $Y : [a,b] \to \mathbb{R}$ and $X : [a,b] \to \mathbb{R}$ be two paths. The **path integral of** Y_t **against** X_t is defined as*

$$\int_a^b Y_t dX_t = \int_a^b Y_t \dot{X}_t dt. \tag{3.1.2}$$

Note that imposing $Y_t = f(X_t)$ in the equation (3.1.2) we recover the equation (3.1.1).

3.1.2 The Signature of a Path

Let $X : [a,b] \to \mathbb{R}^d$ be a path with $X_t = \{X_t^1, X_t^2, \dots, X_t^d\}$ (all X^i are paths from the interval $[a,b]$ to $\mathbb{R}$). We define for every index $i \in \{1, \dots, d\}$

$$S(X)_{a,t}^i = \int_{a<s<t} dX_s^i = X_t^i - X_0^i. \tag{3.1.3}$$

Equation (3.1.3) can be seen as the increment of the i-th coordinate from time 0 to time $t \in [a,b]$. Note that $S(X)_{a,t}^i$ is also a path defined as X_i (meaning that is also defined from the interval $[a,b]$ to $\mathbb{R}$).

Definition 3.1.5. *For every $i, j \in \{1, \dots, d\}$ we define the **double-iterated integral** as*

$$S(X)_{a,t}^{i,j} = \int_{a<s<t} S(X)_{a,s}^i dX_s^j = \int_{a<r<s<t} dX_r^i dX_s^j \tag{3.1.4}$$

The result of equation (3.1.4) is a special case of the path integral defined in (3.1.1) so $S(X)_{a,\cdot}^{i,j} : [a,b] \to \mathbb{R}$ is also a real-valued path. Now, we can define analogously the triple-iterated integral.

Definition 3.1.6. *For every $i, j, k \in \{1, \dots, d\}$ we define the **triple-iterated integral** as*

$$S(X)_{a,t}^{i,j,k} = \int_{a<s<t} S(X)_{a,s}^{i,j} dX_s^k = \int_{a<q<r<s<t} dX_q^i dX_r^j dX_s^k. \tag{3.1.5}$$

The same principle applies as with the double-iterated integral. It's a special case of the path integral, where $S(X)_{a,\cdot}^{i,j,k} : [a,b] \to \mathbb{R}$ represents a real-valued path. Continuing recursively, we obtain the k-fold iterated integral.

Definition 3.1.7. *For any integer $k \geq 1$ and a set of indices $i_1, \dots, i_k \in \{1, \dots, d\}$, we define the* ***k-fold iterated integral*** *of X along the indices $i_1, \dots, i_k \in \{1, \dots, d\}$ as*

$$S(X)_{a,t}^{i_1,\dots,i_k} = \int_{a<s<t} S(X)_{a,s}^{i_k\dots,i_{k-1}} dX_s^{i_k} = \int_{a<t_k<t} \cdots \int_{a<t_1<t_2} dX_{t_1}^{i_1} \dots dX_{t_k}^{i_k}. \tag{3.1.6}$$

Another time, $S(X)_{a,t}^{i_1,\dots,i_k}$ and $X_s^{i_k}$ are real-valued paths, equation (3.1.6) is a special case of the path integral and $S(X)_{a,\cdot}^{i_1,\dots,i_k}$ is a path from the interval $[a, b]$ to $\mathbb{R}$.

Before defining the signature we need to introduce one last concept. Let A be an alphabet of d letters, this is $A = \{1, \dots, d\}$. We define the set words W of the alphabet A as the set

$$W = \{(i_1, \dots, i_k) | k \geq 1, i_1, \dots, i_k \in A\}.$$

Finally we can define the signature of a path.

Definition 3.1.8. *One can compute the* ***signature*** *of a path $X : [a, b] \to \mathbb{R}^d$ (noted as $S(X)_{a,b}$) by computing the collection of all iterated integrals of X. By convention, the first term of the signature is always 1. Rigorously,*

$$S(X)_{a,b} = (1, S(X)_{a,b}^1, \dots, S(X)_{a,b}^d, S(X)_{a,b}^{1,1}, S(X)_{a,b}^{1,2}, \dots), \tag{3.1.7}$$

where the superscripts of the components of $S(X)_{a,b}$ are the set of words W on the alphabet $A = \{1, \dots, d\}$.

Since the words W on an alphabet A are infinite, note that the signature of a path is an infinite series. For this reason sometimes it is convenient to use the k-th level of the signature. This is the finite set of all the $S(X)_{a,b}^{i_1,\dots,i_k}$ where the superscripts have a length of k. For instance, the first level of the signature is $(S(X)_{a,b}^1, \dots, S(X)_{a,b}^d)$, the second level is

$$(S(X)_{a,b}^{1,1}, \dots, S(X)_{a,b}^{1,d}, S(X)_{a,b}^{2,1}, \dots S(X)_{a,b}^{2,d}, \dots, S(X)_{a,b}^{d,1}, \dots S(X)_{a,b}^{d,d})$$

and so on up to k-th level.

3.1.3 Geometrical interpretation of the firsts levels of the Signature

To gain a deeper understanding, let's explore the geometric intuition behind the first two levels of the signature. In equation (3.1.3) we have already seen that the first level of the signature, $S(X)_{a,b}^i$, is the increment of the i-th coordinate (with $i \in \{1, \dots, d\}$). For the second level we will differentiate two cases:

- the first case is when the two superscripts are the same value, this is $S(X)^{i,i}_{a,b}$. With simple computations we obtain for all $i \in \{1, \dots, d\}$:

$$\begin{aligned} S(X)^{i,i}_{a,b} &= \int_a^b \left(\int_a^s dX^i(r) \right) dX^i(s) \\ &= \int_a^b \left(\int_r^b dX^i(s) \right) dX^i(r) \\ &= \int_a^b (X^i(b) - X^i(r)) dX^i(r) \\ &= X^i(b) \int_a^b dX^i(r) - \int_a^b X^i(r) dX^i(r) \\ &= X^i(b)(X^i(b) - X^i(a)) - \left(\frac{(X^i(b))^2}{2} - \frac{(X^i(a))^2}{2} \right) \\ &= \frac{(X^i_b - X^i_a)^2}{2}. \end{aligned}$$

- The second case is when the superscripts are $i, j \in \{1, \dots, d\}$ for $i \neq j$. The terms $S(X)^{i,j}_{a,b}$ have something to do with the Lévy area. The Lévy area, represented in Figure 3.2, refers to the signed area enclosed by the path, which is the red line, and the chord, depicted by the blue dashed straight line connecting the endpoints. The area above the blue chord (A^-) is negative and the one below the chord (A^+) is positive, this is the reason why we say that it is a signed area. For a two dimensional path $X_t = \{X^1_t, X^2_t\}$ the Lévy area is

$$A = \frac{1}{2} \left(S(X)^{1,2}_{a,b} - S(X)^{2,1}_{a,b} \right).$$

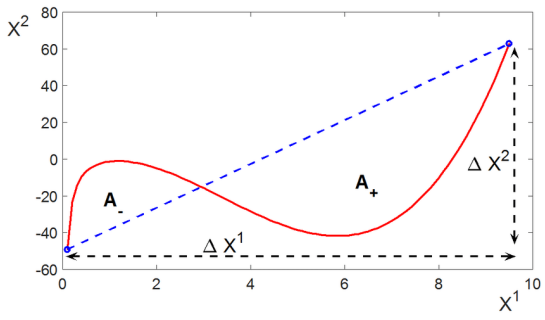


FIGURE 3.2: Lévy area, image extracted from Chevyrev and Kormilitzin, 2016

3.1.4 Embedding: from discrete data to a path

Understanding how to construct a path from discrete data is crucial, especially in time series analysis where data points are often discrete due to the discrete nature of time variables. There are three primary methods for constructing such paths: piecewise linear interpolation, rectilinear interpolation, and lead-lag transformation.

Piecewise linear interpolation is a method used to estimate values between known data points by approximating each segment between adjacent points with a straight line. This technique assumes linearity between neighboring data points, providing

a simple yet effective method for filling in missing or intermediate values in one-dimensional datasets. We will illustrate this with an example.

Example 3.1.9. Let X and t be one-dimensional data streams of length four defined by

$$\{X_i\}_{i=1}^4 = \{1,5,3,6\} \quad \text{and} \quad \{t_i\}_{i=1}^4 = \{0,1,2,3\}.$$

Using the piecewise linear interpolate we transform the discrete series into a fucntion or path. We can see the representation of the picewise linear interpolation in Figure 3.3. Note that we just joined the data points with a straight segment.

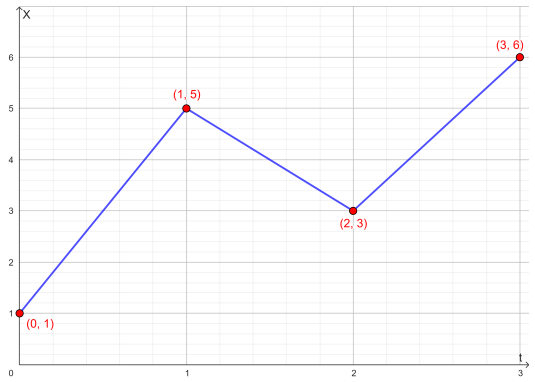


FIGURE 3.3: Piecewise linear interpolation

The next method is the rectilinear interpolation. Rectilinear interpolation is a method used to estimate values within a rectangular grid of known data points. Given a grid of data points arranged in "rows" and "columns", rectilinear interpolation estimates the value at a point within the grid by interpolating along the rows and columns independently. In other words, it first performs linear interpolation along one direction (e.g., horizontally) to estimate values along the rows, and then performs another linear interpolation along the other direction (e.g., vertically) to estimate values along the columns. We will use the same data that we used in Example 3.1.9 to see how this method works.

Example 3.1.10. Let X and t be the same as in the previous example. We can construct the rectilinear interpolation by adding the auxiliary points $(1,1)$, $(2,5)$ and $(3,3)$ that are represented as orange crosses in Figure 3.4.

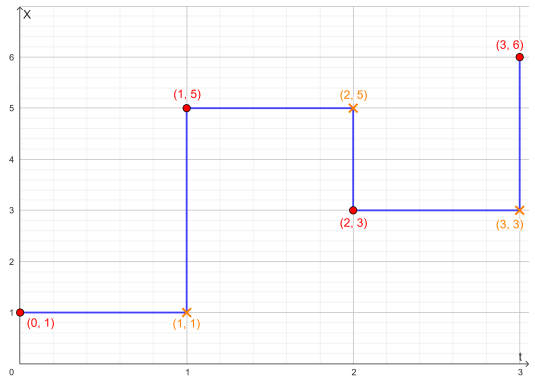


FIGURE 3.4: Rectilinear interpolation

Finally we will present the lead-lag transformation. Lead-lag transformation is fundamentally different from interpolation techniques. It involves shifting one time series $\{X_t\}$ relative to another $\{Y_t\}$ in order to analyze the temporal relationships between variables. The idea is to shift one time series relative to another, either forward (lead) or backward (lag) in time. The process $\{X_t\}$ follows $\{Y_t\}$'s path with a k lag:

$$X_{t+k} \propto Y_t, \quad k > 0.$$

This transformation reveals dynamic dependencies and correlations, making it particularly useful in time series analysis and financial modeling. Let us show the process of computing the lead-lag transformation with an example.

Example 3.1.11. Let X be the same as in the two previous examples. We want to compute the lead-lag transformation for the exact same process X:

$$\text{Lead-lag} : X = \{1,5,3,6\} \mapsto \begin{cases} X^{Lag} & = \{1,5,5,3,3,6,6\} \\ X^{Lead} & = \{1,1,5,5,3,3,6\} \end{cases}$$

The resulting path is represented in Figure 3.5. The orange crosses represent the auxiliary points added, which are: $(1,5)$, $(5,3)$ and $(3,6)$.

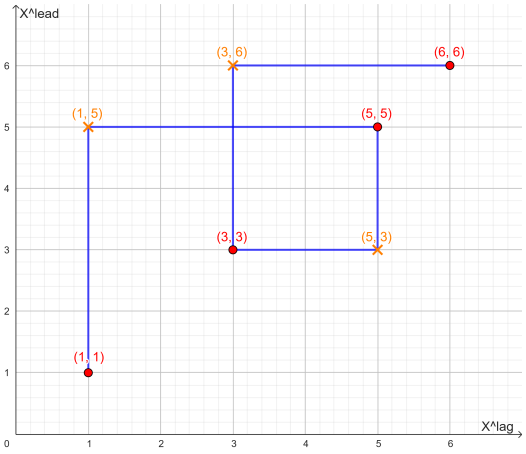


FIGURE 3.5: Lead-lag transformation

3.1.5 Signature Properties

Invariance to time re-parameterisation

One of the key properties of signatures is their invariance to time re-parameterisation. This means that the signature of a path remains unchanged under a continuous and strictly increasing transformation of time. To understand this, consider a continuous path X defined on the interval $[0,T]$.

Suppose we have a continuous and strictly increasing function $\sigma : [0,T] \to [S,U]$. This function re-parameterises the time interval $[0,T]$ to $[S,U]$. For any multi-index $(i_1,\ldots,i_k) \in I$ and for any $0 \leq s < t \leq T$, the invariance property can be expressed as:

$$\int_{s<u_1<\cdots<u_k<t} dX^{i_1}_{u_1} \cdots dX^{i_k}_{u_k} = \int_{\sigma(s)<u_1<\cdots<u_k<\sigma(t)} dX^{i_1}_{\sigma^{-1}(u_1)} \cdots dX^{i_k}_{\sigma^{-1}(u_k)}.$$

Uniqueness

The uniqueness property of signatures underscores their power in characterizing paths. As discussed in Hambly and Lyons, 2010, the signature of a path $S_{s,t}(X)$ uniquely determines the function $u \mapsto X_u - X_s$ for $u \in [s,t]$ up to tree-like equivalence. This means the signature encodes sufficient information to reconstruct the path segment X_u relative to X_s within $[s,t]$.

A sufficient condition for this uniqueness is the existence of a component $i \in \{1,\dots,d\}$ such that X_u^i is strictly monotone increasing, ensuring the path does not retrace itself and can be distinguished by its signature.

Typically, the first few terms of the signature contain most of the path's information, making signatures efficient representations. This is especially useful in practical applications, providing a concise yet comprehensive summary of the path's behavior.

Log Signature

A formal power series x is expressed as:

$$x = \sum_{k=0}^{\infty} \sum_{i_1,\dots,i_k} \lambda_{i_1,\dots,i_k} e_{i_1} \otimes \cdots \otimes e_{i_k}$$

where $\lambda_{i_1,\dots,i_k}$ are coefficients and $e_{i_1},\dots,e_{i_k}$ are basis elements in a vector space.

The logarithm of x, denoted $\log x$, is defined by:

$$\log x = \log(\lambda_0) + \sum_{n\geq 1} \frac{(-1)^n}{n} \left(1 - \frac{x}{\lambda_0}\right)^{\otimes n}$$

where $\lambda_0 > 0$ ensures $\log(\lambda_0)$ is well-defined, and $\otimes$ denotes the n-th tensor power (for more information about tensor algebra see Akivis, Goldberg, and Silverman, 1972).

For a path $X : [a,b] \to \mathbb{R}^d$, the log signature $\log S(X)_{a,b}$ is the formal power series obtained by applying the logarithmic transformation to the signature $S(X)_{a,b}$ over the interval $[a,b]$.

It is important to define the Lie bracket between two formal power series x and y. We define the Lie bracket as:

$$[x,y] = x \otimes y - y \otimes x$$

This operation analyzes algebraic structures and commutative properties within formal power series.

Now, the first few terms of $\log S(X)_{a,b}$ are:

$$\log S(X)_{a,b} = \sum_{i=1}^{d} S(X)_{a,b}^{i} e_i + \sum_{1\leq i<j\leq d} \frac{1}{2} \left(S(X)_{a,b}^{i,j} - S(X)_{a,b}^{j,i}\right) [e_i, e_j] + \dots$$

Here, $S(X)_{a,b}^{i}$ denotes the i-th component of $S(X)_{a,b}$, and $S(X)_{a,b}^{i,j}$ represents the Lévy area between e_i and e_j.

In summary, the log signature refines path representations by capturing complex interactions, enhancing interpretability in sequential data analysis such as pattern recognition and anomaly detection in data science applications.

3.1.6 Properties of the signature of a time series

In this section, we discuss the main properties of the signature of a time series $\{(t_i, r_i)\}_{i=m}^{N}$ through the presentation of two important lemmas.

3.1.7 Unique Determination by Signature

Lemma 3.1.12 (Unique Determination by Signature)**.** *Suppose that* $0 < m < n \leq N$*, where* $m, n \in \mathbb{N}$*. The signature of a time series* $\{(t_i, r_i)\}_{i=m}^{n}$ *uniquely determines the time series* $\{(t_i, r_i)\}_{i=m}^{n}$*.*

Lemma 3.1.12 states that the signature of a time series encodes enough information to uniquely reconstruct the original time series. In other words, given the signature, we can accurately determine the sequence of time points and their corresponding values. The proof can be found in Levin, Lyons, and Ni, 2016

Lemma 3.1.13 (Linear Functional Representation)**.** *Let* $\boldsymbol{X}$ *denote the signature of a time series* $\{(t_i, r_i)\}_{i=1}^{n}$*. Assume that the time points* $\{t_i\}_{i=1}^{n}$ *are known. Then the differences in the values* $\Delta R = (r_1, r_2 - r_1, \ldots, r_n - r_{n-1})^T$ *can be represented as a linear functional of the signature* X*, specifically:*

$$\Delta R = T^{-1} A,$$

where

$$A := \begin{pmatrix} 0!\pi^2(\boldsymbol{X}) \\ 1!\pi^{12}(\boldsymbol{X}) \\ \vdots \\ (n-1)!\pi^{1\ldots12}(\boldsymbol{X}) \end{pmatrix}, \quad T := \begin{pmatrix} 1 & 1 & \cdots & 1 \\ t_1 & t_2 & \cdots & t_n \\ \vdots & \vdots & \ddots & \vdots \\ t_1^{n-1} & t_2^{n-1} & \cdots & t_n^{n-1} \end{pmatrix}, \quad \Delta R := \begin{pmatrix} r_1 \\ r_2 - r_1 \\ \vdots \\ r_n - r_{n-1} \end{pmatrix}.$$

Here, the notation π refers to the projection of the signature $\mathbf{X}$ onto specific components. Specifically, $\pi^2(\mathbf{X})$ denotes the second component of the signature of X, $\pi^{12}(\mathbf{X})$ denotes the projection onto the component corresponding to the indices 1 and 2, and so forth. The proof of this lemma is in Levin, Lyons, and Ni, 2016.

Lemma 3.1.13 demonstrates how the values of a time series can be derived from its signature when the time points are known. By expressing ΔR as $T^{-1}A$, we establish a linear relationship between the time series values and its signature. This means that once we have the signature and the time points, we can use this linear functional to retrieve the original time series values. This property is particularly useful in data science applications where efficient and accurate reconstruction of time series data is needed for analysis, forecasting, or anomaly detection.

Chapter 4

Application 1: Signature Method for Model Classification

In this section, we will illustrate an application of the signature method. Specifically, we will simulate two models, Y and Y', which have the same structure but different parameter values, and then classify them using the signature of the time series. We will test it with the following model structures: $AR(p)$, $ARMA(p,q)$ and $ARIMA(p,d,q)$.

4.1 Experiments and results

4.1.1 AR(p) model

Consider two $AR(p)$ models that share the same order p but have different parameters: Φ_p or Φ'_p. Let $\{Y_{i,j}\}$ represent the observations of the time series, where the index i denotes each individual time series and j represents the observation at time t_j. Each time series $Y_{i,\cdot}$ is assigned a label: "class 0" or "class 1", depending on the parameters used to generate the time series. In this case we will fix $p = 2$ and the classes will be defined by:

- class 0: $Y_t = -0.5Y_{t-1} + 0.25Y_{t-2} + Z_t$ and
- class 1: $Y_t = -0.75Y_{t-1} + 0.5Y_{t-2} + Z_t$,

where Z_t has zero mean and unit variance. We will generate 500 time series of each class, 1000 in total, of length 100.

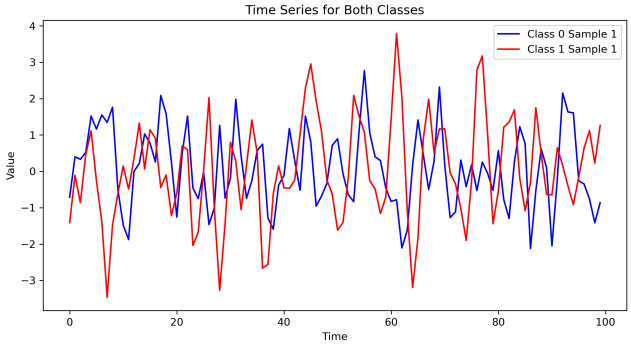


FIGURE 4.1: Two AR(2) series of different classes

In Figure 4.1 we can see two examples of the time series generated. The blue one is defined by the class 0 scheme and the red one corresponds to class 1. After generating the 1000 series we store the series in a matrix with this structure:

t_1	t_2	t_3	...	t_{100}	**Class**
$Y_{1,1}$	$Y_{1,2}$	$Y_{1,3}$	...	$Y_{1,100}$	0
$Y_{2,1}$	$Y_{2,2}$	$Y_{2,3}$	...	$Y_{2,100}$	0
$Y_{3,1}$	$Y_{3,2}$	$Y_{3,3}$	...	$Y_{3,100}$	0
⋮	⋮	⋮	⋱	⋮	⋮
$Y_{500,1}$	$Y_{500,2}$	$Y_{500,3}$	...	$Y_{500,100}$	0
$Y_{1,1}$	$Y_{1,2}$	$Y_{1,3}$	...	$Y_{1,100}$	1
$Y_{2,1}$	$Y_{2,2}$	$Y_{2,3}$	...	$Y_{2,100}$	1
$Y_{3,1}$	$Y_{3,2}$	$Y_{3,3}$	...	$Y_{3,100}$	1
⋮	⋮	⋮	⋱	⋮	⋮
$Y_{500,1}$	$Y_{500,2}$	$Y_{500,3}$	...	$Y_{500,100}$	1

Then we are going to transform each $\{Y_i\}$. Firstly, we use the cumulative sum:

$$\tilde{Y}_{ij} = CS(\{Y_i\}_j) = \{0, Y_{i,1}, Y_{i,1} + Y_{i,2}, Y_{i,1} + Y_{i,2} + Y_{i,3}, \dots\}$$

and later we convert each time series in a two dimensional path by using the lead-lag transformation obtaining

$$X_i = \left\{ \left(\tilde{Y}_i^{\text{lead}}, \tilde{Y}_i^{\text{lag}} \right) \right\}.$$

After this transformation, we are able to compute the signature of each time series Y_i that has been converted into a path. To compute the signature in Python we use the `esig` library. According to PyPI, 2024 "The Python package esig provides a toolset (previously called sigtools) for transforming vector time series in stream space to signatures in effect space". We are going to compute the signature up to level $L = 2$, this is:

$$S(X_i) = \left(1, S_i^{(1)}, S_i^{(2)}, S_i^{(1,1)}, S_i^{(1,2)}, S_i^{(2,1)}, S_i^{(2,2)}\right)$$

And then store all the signatures in a matrix with form:

Feature Set						**Class**
$\hat{S}_1^{(1)}$	$\hat{S}_1^{(2)}$	$\hat{S}_1^{(1,1)}$	$\hat{S}_1^{(1,2)}$	$\hat{S}_1^{(2,1)}$	$\hat{S}_1^{(2,2)}$	0
$\hat{S}_2^{(1)}$	$\hat{S}_2^{(2)}$	$\hat{S}_2^{(1,1)}$	$\hat{S}_2^{(1,2)}$	$\hat{S}_2^{(2,1)}$	$\hat{S}_2^{(2,2)}$	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{S}_{500}^{(1)}$	$\hat{S}_{500}^{(2)}$	$\hat{S}_{500}^{(1,1)}$	$\hat{S}_{500}^{(1,2)}$	$\hat{S}_{500}^{(2,1)}$	$\hat{S}_{500}^{(2,2)}$	0
$\hat{S}_1^{(1)}$	$\hat{S}_1^{(2)}$	$\hat{S}_1^{(1,1)}$	$\hat{S}_1^{(1,2)}$	$\hat{S}_1^{(2,1)}$	$\hat{S}_1^{(2,2)}$	1
$\hat{S}_2^{(1)}$	$\hat{S}_2^{(2)}$	$\hat{S}_2^{(1,1)}$	$\hat{S}_2^{(1,2)}$	$\hat{S}_2^{(2,1)}$	$\hat{S}_2^{(2,2)}$	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\hat{S}_{500}^{(1)}$	$\hat{S}_{500}^{(2)}$	$\hat{S}_{500}^{(1,1)}$	$\hat{S}_{500}^{(1,2)}$	$\hat{S}_{500}^{(2,1)}$	$\hat{S}_{500}^{(2,2)}$	1

Note that we have removed the first element of the signature in all the rows. In addition, the notation $\hat{S}_i^I$ refers to the signature value standardized for row $i =$

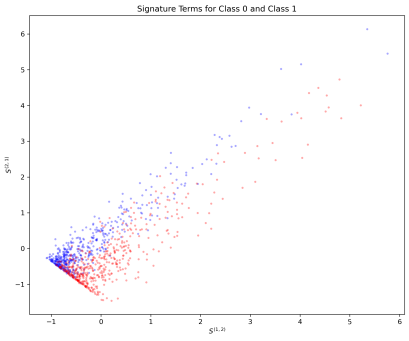
$1, \ldots, 1000$ and column $I \in \{(1), (2), (1,1)(1,2), (2,1), (2,2)\}$. To standarize the each S_i^I we compute the mean of the I-th column μ_I, and the standard deviation of the I-th column σ_I by using the formulas

$$\mu_I = \frac{1}{1000} \sum_{i=1}^{1000} S_i^I \quad \text{and} \quad \sigma_I = \sqrt{\frac{1}{1000} \sum_{i=1}^{1000} (S_i^I - \mu_I)^2},$$

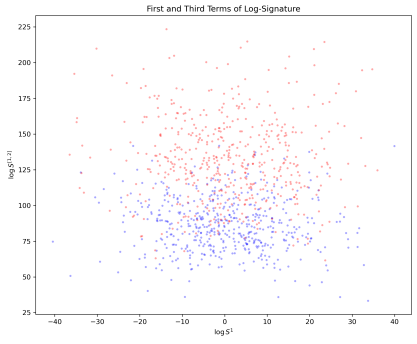
respectively. Then the values of the standardized matrix are given by:

$$\hat{S}_i^I = \frac{S_i^I - \mu_I}{\sigma_I}.$$

The objective is to predict the corresponding class label for each row i, given $\hat{S}_i = (\hat{S}_1^{(1)}, \ldots, \hat{S}_1^{(2,2)})$. We will follow the standard procedure of splitting the data into a training set and a testing set. The testing set comprises 30% of the data, while the remaining data is used for training. Before training a logistic regression model to directly predict whether the given data belongs to class 0 or 1, we will first explore why the first and second levels of the signatures (in this case) are appropriate for classifying the data. With the training data, we will train a LASSO model to determine the most relevant signatures for predicting the class. The results of the analysis indicate that the most significant signatures are $(1,2)$ and $(2,1)$, with corresponding LASSO coefficients of 0.48842699 and -0.74945231, respectively. In Figure 4.2a we have plotted the signature term $S^{(1,2)}$ in the x axis and $S^{(2,1)}$ in the y axis. We have visually distinguished the classes of the data by using different colors. One can see that this shows a separation on data. Furthermore, we have plotted in Figure 4.2b $\log S^{(1)}$ against $\log S^{[1,2]}$ to provide additional insight into the data separation.



(A) Figure 4.2: Relevant signature terms for class 0 and class 1



(B) Figure 4.2: Log signature terms for class 0 and class 1

After that, we discard the $S^{(I)}$ with LASSO coefficients equal to zero and train a logistic regression model. On the left we can see the confusion matrix for the training dataset with an accuracy of 0.8385 and on the right the confusion matrix corresponding to the test set with also an accuracy of 0.8167.

Training Data

Actual	Predicted 0	Predicted 1
Class 0	300	50
Class 1	63	287

Accuracy: 0.8386

TABLE 4.1: Training data results

Testing Data

Actual	Predicted 0	Predicted 1
Class 0	148	2
Class 1	13	137

Accuracy: 0.8167

TABLE 4.2: Testing data results

Overall, the model demonstrates good performance on both the training and testing datasets, with high accuracies. This indicates that the selected feature matrix is effective in distinguishing between the classes. Subsequently, we will examine the same process for another model.

4.1.2 ARMA(p,q) model

In this section, the objective is to apply the same methodology as previously discussed to analyze the effectiveness of time series signatures in classifying them. In this instance, the time series will adhere to an ARMA(1,1) process. Two classes will be defined, as in the previous experiment. The equations representing the classes are as follows:

- class 0: $Y_t - 0.4Y_{t-1} = 0.5 + Z_t + 0.5Z_{t-1}$ and
- class 1: $Y_t - 0.8Y_{t-1} = 0.5 + Z_t + 0.7Z_{t-1}$.

In continuation with our methodology from the previous experiment, we will omit a detailed explanation of the entire process. Instead, we will focus on plotting key elements and analyzing the outcomes. In Figure 4.3, we present the plot of two ARMA(1,1) processes, with colors indicating their respective classes.

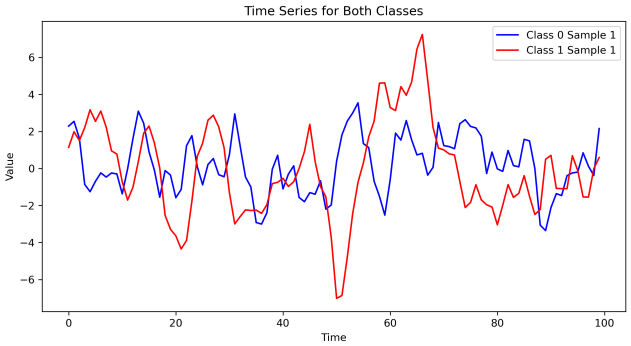
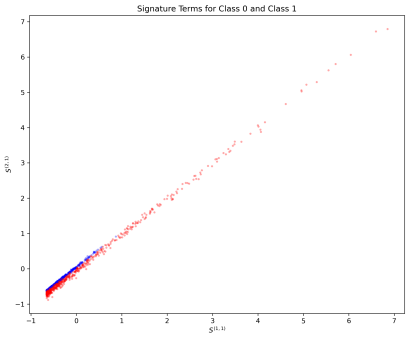


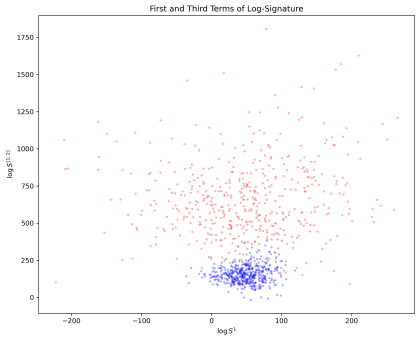
FIGURE 4.3: Two ARMA(1,1) series of different classes

The LASSO model identified the most relevant signatures as those with superscripts $(1,1)$ and $(2,1)$. In Figure 4.4a, we present these signatures plotted against each other. Additionally, in Figure 4.4b, we plot $\log S^{(1)}$ against $\log S^{[1,2]}$. Both representations clearly illustrate the separation between the two classes.

Here are the confusion matrices and accuracies for both the training and testing sets:



(A) Figure 4.4: Relevant signature terms for class 0 and class 1



(B) Figure 4.4: Log signature terms for class 0 and class 1

Actual	Predicted 0	Predicted 1
Class 0	349	1
Class 1	20	330

Accuracy: 0.97

TABLE 4.3: Training Data

Actual	Predicted 0	Predicted 1
Class 0	148	2
Class 1	13	137

Accuracy: 0.95

TABLE 4.4: Testing Data

The model performs exceptionally well using the selected feature matrix, effectively classifying the data with high accuracy and low error rates. The high accuracies on both training (0.97) and testing data (0.95) suggests that the features selected are highly relevant and informative for the classification task.

4.1.3 ARIMA(p,d,q)

We employ the same approach for an ARIMA (2,1,2) model. The classes are defined as follows:

- class 0: $(1 - 0.5B + 0.25B^2)(1 - B)Y_t = (1 + 0.3B - 0.2B^2)Z_t$ and
- class 1: $(1 - 0.75B + 0.5B^2)(1 - B)Y_t = (1 + 0.4B - 0.3B^2)Z_t$

In Figure 4.5, we illustrate the graphical representations of two ARIMA(1,2,1) processes, where colors distinguish between their respective classes.

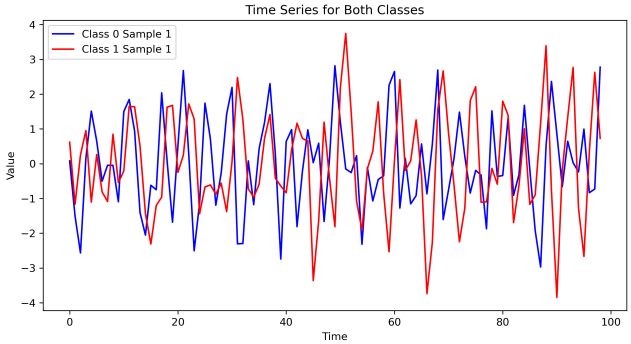
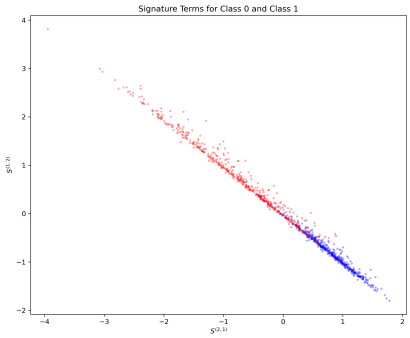
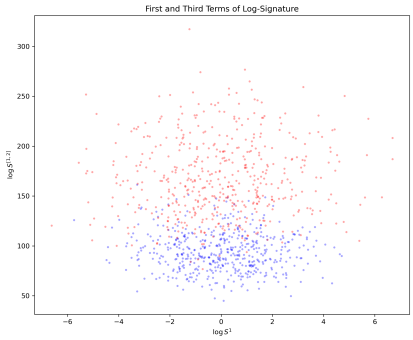


FIGURE 4.5: Two ARMA(1,1) series of different classes

The LASSO analysis highlighted the significance of signatures denoted as $(2,1)$ and $(1,2)$. In Figure 4.6a, we plot these signatures for visual comparison. Furthermore, in Figure 4.6b, we show $\log S^{(1)}$ against $\log S^{[1,2]}$, revealing a distinct demarcation between the classes.



(A) Figure 4.6: Relevant signature terms for class 0 and class 1



(B) Figure 4.6: Log signature terms for class 0 and class 1

Following is the presentation of confusion matrices and accuracy metrics for the training and testing sets subsequent to training a logistic regression model with LASSO feature selection.

Actual	Predicted 0	Predicted 1
Class 0	323	27
Class 1	40	310

Accuracy: 0.9042

TABLE 4.5: Training Data

Actual	Predicted 0	Predicted 1
Class 0	138	12
Class 1	17	133

Accuracy: 0.9033

TABLE 4.6: Testing Data

In the training data, the model achieved a relatively high accuracy of approximately 90.43%. Similarly, on the testing data, the model demonstrated a commendable accuracy of around 90.33%. Overall, these results suggest that the model classifies correctly the data.

Chapter 5

Application 2: Classification of Crude Oil Data Stream Based on 30-Minute Time Intervals

In this chapter, we delve into the application of the signature method for the classification of financial data streams, with a specific focus on the crude oil stock market. The central objective is to classify 30-minute time windows of crude oil stock market data by using its signature. This chapter is based on a paper titles "Extracting information from the signature of a financial data stream" Gyurkó et al., 2014.

Crude oil stock market data, characterized by its high volatility and complex temporal dependencies, presents a significant challenge for traditional classification techniques. The inherent noise and rapid fluctuations in crude oil prices needs a robust analytical framework capable of perceiving subtle patterns over short time intervals. The signature method, with its capacity to handle irregular and high-frequency data, is particularly suited to this task.

Classifying stock market data is crucial for several reasons. Accurate classification can enhance predictive models, allowing traders and analysts to make more informed decisions and develop more effective trading strategies. Moreover, it can help in risk management by identifying potential market shifts and anomalous behaviors promptly. In the context of crude oil, which plays an important role in the global economy, understanding its market dynamics can lead to better insights into macroeconomic trends and energy policies. Therefore, the ability to classify and interpret crude oil market data accurately is fundamental for financial stability and strategic planning.

This approach represents a novel application of the signature method. Previously, we classified simulated time series where the underlying model types were known. In contrast, we will now apply this method to a real financial data stream without detailed knowledge of the data's nature. This shift to using actual market data presents new challenges and opportunities, highlighting the robustness and versatility of the signature method in handling complex, real-world financial data.

5.1 Crude Oil Market Data: Analysis and Preprocessing

To conduct our experiments, we will utilize a dataset comprising FX prices of crude oil, provided in generic ASCII format, denominated in WTI/USD (West Texas Intermediate in USD). The dataset consists of 1-minute interval data, recorded in Eastern

Standard Time (EST) zone without Daylight Savings adjustments. Each data point within the 1-minute window includes the following columns:

- Timestamp (T): Indicates the specific time at which the price data was recorded.
- Open Price (O): Refers to the price of crude oil at the beginning of the 1-minute interval.
- High Price (H): Represents the highest price of crude oil reached during the 1-minute interval.
- Low Price (L): Denotes the lowest price of crude oil recorded within the 1-minute interval.
- Close Price (C): Signifies the price of crude oil at the end of the 1-minute interval.
- Volume (V): represents the total amount of trading activity.

The column "volume" will be disregarded as it consistently registers a value of 0. The dataset spans the years 2021, 2022, and 2023. The data has been obtained from HistData.com, 2024.

We will generate data streams spanning 30-minute intervals, starting either from minute 00 to 29 or from minute 30 to 59. Subsequently, we will preprocess these data streams represented as $\hat{Y} = ((T, O, H, L, C)_{t_i})_{i=1}^{30}$. To mitigate the presence of obvious patterns, we will apply various transformations, including normalization and standardization, to the features within the data. The transformations applied are:

- We begin by normalizing the timestamps T. This normalization ensures that the timestamps are represented on a scale from 0 to 1, indicating their position within the time series. The normalization formula is given by:

$$u_{t_i} := \frac{_i - t_0}{t_{30} - t_0}.$$

- Next, we normalize the open, high, low, and close prices by subtracting the mean μ and dividing by the standard deviation σ of each respective column. This standardization process ensures that each feature has a mean of 0 and a standard deviation of 1, allowing for better comparison and analysis. The formulas for normalization are as follows:

$$o_{t_i} := \frac{O_{t_i} - \mu(\{O_{t_i}\}_{i=0}^{30})}{\sigma(\{O_{t_i}\}_{i=0}^{30})}, \qquad h_{t_i} := \frac{H_{t_i} - \mu(\{H_{t_i}\}_{i=0}^{30})}{\sigma(\{H_{t_i}\}_{i=0}^{30})},$$

$$l_{t_i} := \frac{L_{t_i} - \mu(\{L_{t_i}\}_{i=0}^{30})}{\sigma(\{L_{t_i}\}_{i=0}^{30})} \quad \text{and} \quad c_{t_i} := \frac{C_{t_i} - \mu(\{C_{t_i}\}_{i=0}^{30})}{\sigma(\{C_{t_i}\}_{i=0}^{30})}.$$

- Additionally, we calculate the logarithm of the mid price (m), which is the average of the high and low prices. This transformation provides a more symmetric representation of the price distribution and helps capture potential patterns in the data. The formula for calculating the logarithm of the mid price is:

$$mp_{t_i} := \log \frac{H_{t_i} + L_{t_i}}{2}.$$

- Furthermore, we compute the spread between the high and low prices and standardize it by subtracting the mean and dividing by the standard deviation. This standardized spread provides a measure of price volatility relative to the mean spread across the dataset. The formula for calculating the standardized spread is:

$$s_{t_i} := \frac{S_{t_i} - \mu(\{S_{t_i}\}_{i=0}^{30})}{\sigma(\{S_{t_i}\}_{i=0}^{30})},$$

  where $S_{t_i} = H_{t_i} - L_{t_i}$.

Now, we have transformed each data stream $\hat{Y}$ into $X = (u_{t_i}, o_{t_i}, h_{t_i}, l_{t_i}, c_{t_i}, mp_{t_i}, s_{t_i})_{i=1}^{30}$. The reference paper suggests implementing a lead-lag transformation by introducing a lag based on the logarithm of the mid-price (mp). However, during our experimentation, this transformation showed computational challenges, primarily due to the complexity of the resulting path and the significant computational time required for signature computations. As an alternative approach, we explored the use of piecewise linear interpolation. Remarkably, this strategy yielded excellent results while consuming minimal computational resources. Therefore, we utilized the `esig` library in Python to calculate the signature of the piecewise linear interpolation paths for all X up to level 4. This results into 2801 signature terms for each data stream.

5.1.1 Classification method

In our experiments, the input comprises a collection of data streams spanning two distinct time windows, each lasting 30 minutes, denoted as "time 1" and "time 2". Each data stream is followed by a label indicating its temporal partition. Our objective is to leverage the unique signature of each data stream to predict whether it belongs to time 1 or time 2.

To accomplish this, we divide the data into separate training and testing sets. The testing set constitutes 25% of the total input data streams. The prediction task involves assigning a value of 0 to data streams corresponding to time 1 and a value of 1 to those from time 2.

For classification, we employ a linear regression model augmented with Lasso regularization, setting the hyperparameter alpha to 0.01 (by k-fold cross validation it always resulted to be the best option). Standardization of the signature features is performed to ensure optimal functioning of the LASSO regularization technique.

5.1.2 Performance evaluation

To evaluate the performance of our classification method, we employ the Kolmogorov-Smirnov (KS) distance. The KS statistic measures the maximum distance between the cumulative distribution functions (CDFs) of two samples, ranging from 0 to 1. Higher KS distances indicate better separation between the positive and negative classes, meaning the model is more effective at distinguishing between the two classes.

We begin by computing the KS distance for the distributions of the predicted probabilities for each category using both the training set and the test set. This involves calculating the KS distance for the predicted probabilities of the positive and

negative classes in both datasets, providing insight into the model's ability to differentiate between the two classes.

Next, we use a threshold of 0.5 to classify instances into positive and negative classes. We then compute the confusion matrices for the training and testing sets, allowing us to calculate the ratio of correct classifications (accuracy) for both sets. This step helps us evaluate the model's performance in terms of correctly identifying positive and negative instances.

Finally, we compute the Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) for both the training and testing sets. The ROC curve illustrates the model's ability to distinguish between classes at threshold value 0.5, while the AUC provides a single scalar value summarizing the overall performance. This comprehensive evaluation process offers a detailed understanding of the model's discriminative power and accuracy across different datasets.

5.2 Experiments and Results

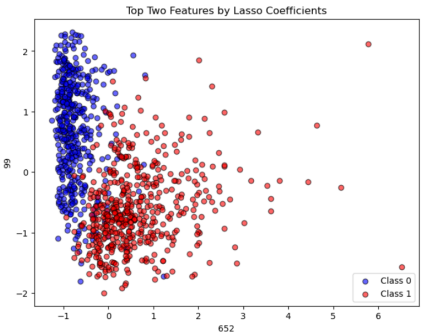
In evaluating market performance, it is essential to consider various time intervals. First, we will compare several segments of the trading day with the final trading window before market closure at 17:00 P.M. to 18:00 P.M. Specifically, we will contrast the market activity during the last time slot of the day, 16:30 P.M. to 16:59 P.M., with the intervals: 9:00 A.M. to 9:29 A.M., 10:30 A.M. to 10:59 A.M., 12:00 P.M. to 12:29 P.M., 14:30 P.M. to 14:59 P.M., 15:00 P.M. to 15:29 P.M. and 16:00 P.M. to 16:29 P.M.. These time windows capture different phases of the trading day, allowing us to analyze market trends and similarities with the closure interval.

In Table 5.1, we observe various performance metrics: KS Distance, AUC and Accuracy, all of which indicate higher values correspond to better classification model performance. Across all cases, we find consistently strong results for these metrics.

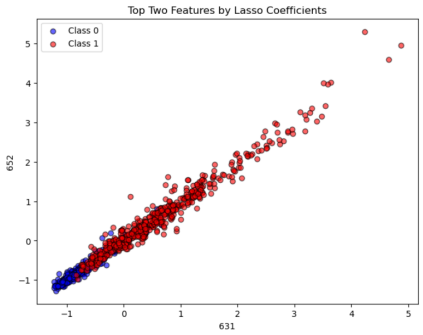
The KS Distance values range from 0.6638 to 0.9401 for the training set and from 0.5472 to 0.8755 for the test set. Regarding AUC, the values range from 0.9060 to 0.9925 for the training set and from 0.8327 to 0.9787 for the test set. Higher AUC values denote better overall performance in distinguishing between classes, as illustrated by the ROC curve. Accuracy, which measures the proportion of correctly classified instances, shows values ranging from 0.7556 to 0.9678 for the training set and from 0.7540 to 0.9346 for the test set. Higher accuracy values indicate a higher percentage of correctly predicted outcomes by the model. An intriguing observation from the data is the trend where earlier time windows (further from market closing time) tend to exhibit better performance across all metrics. As time windows approach closer to the market closing time, the performance metrics tend to decrease, indicating reduced model effectiveness in distinguishing between classes. This trend suggests that temporal proximity to market events significantly impacts the model's predictive performance. One can think that as we move closer to the market closing time, the time windows become more similar in terms of their characteristics or features, for this reason we could have lower performance when we approximate to the closing time.

Window Comparison	KS Distance		AUC		Accuracy	
	Train	Test	Train	Test	Train	Test
9:00-9:29 vs 16:30-16:59	0.9401	0.8755	0.9925	0.9787	0.9678	0.9346
10:30-10:59 vs 16:30-16:59	0.8904	0.8354	0.9869	0.9765	0.9400	0.9068
12:00-12:29 vs 16:30-16:59	0.8122	0.7521	0.9681	0.9404	0.9044	0.8660
14:30-14:59 vs 16:30-16:59	0.8029	0.7602	0.9627	0.9326	0.8983	0.8540
15:00-15:29 vs 16:30-16:59	0.6638	0.5472	0.9060	0.8327	0.8347	0.7556
16:00-16:29 vs 16:30-16:59	0.6713	0.5641	0.9068	0.8506	0.8319	0.7583

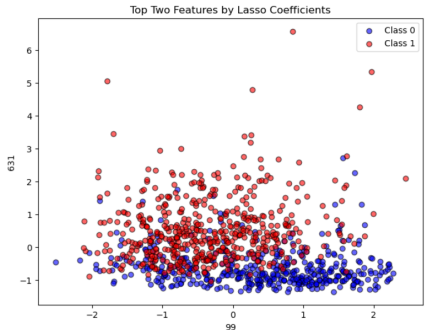
TABLE 5.1: Comparison of Performance Metrics Across Several Time Windows vs Closing Time Window



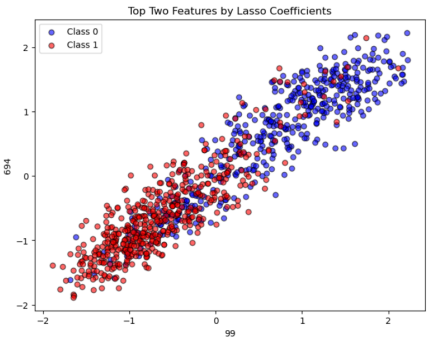
(A) Relevant Signature Terms for 9:00-9:29 vs 16:30-16:59



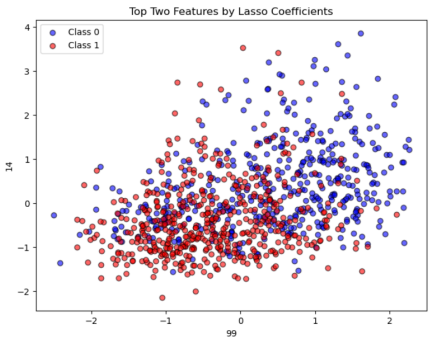
(B) Relevant Signature Terms for 10:30-10:59 vs 16:30-16:59



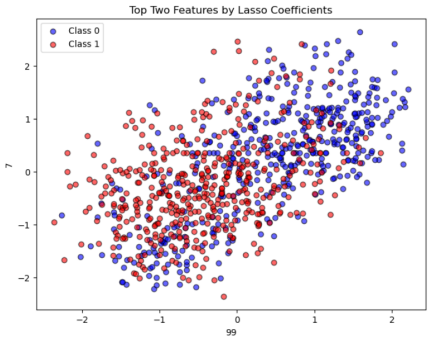
(C) Relevant Signature Terms for 12:00-12:29 vs 16:30-16:59



(D) Relevant Signature Terms for 14:30-14:59 vs 16:30-16:59



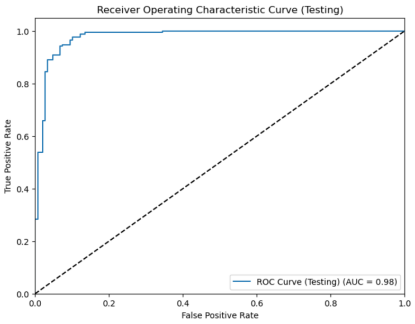
(E) Relevant Signature Terms for 15:00-15:29 vs 16:30-16:59



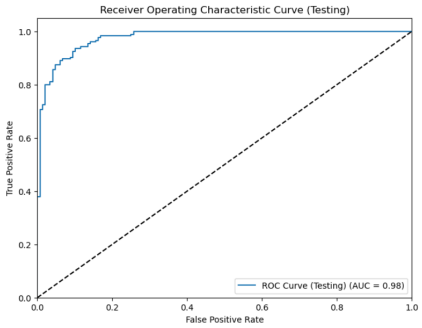
(F) Relevant Signature Terms for 16:00-16:29 vs 16:30-16:59

FIGURE 5.1: Relevant Signature Terms for different Time Windows

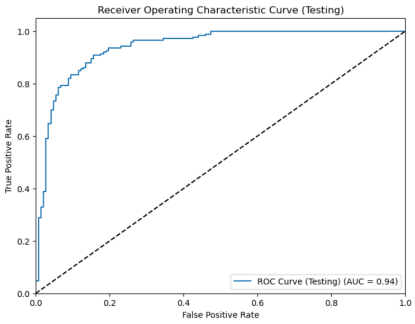
In Figure 5.1, we illustrate a plot showing the relationship between the two most significant signature terms, identified by their highest absolute LASSO coefficients. Class 1 is consistently represented in red and corresponds exclusively to the closing time slot of 16:30 P.M.-16:59 P.M.. Meanwhile, Class 0 varies across the different time slots selected for the experiment. The plot reveals a noticeable trend: as the selected time slots approach the market closing time, the distinction between the classes becomes increasingly ambiguous. This observation suggests that the identified signature terms encounter greater difficulty in effectively partitioning the data as the time slots approach each other.



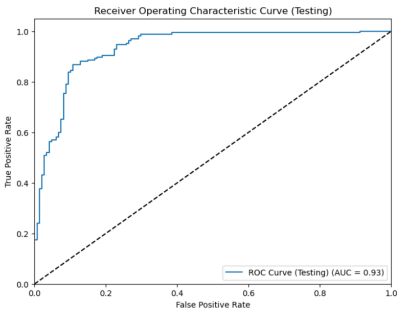
(A) ROC Curve for 9:00-9:29 vs 16:30-16:59



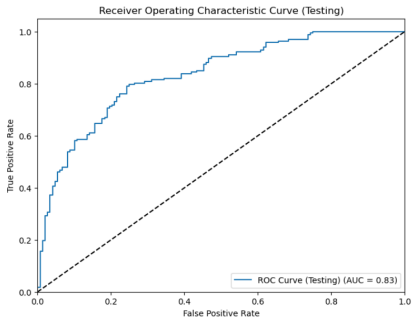
(B) ROC Curve for 10:30-10:59 vs 16:30-16:59



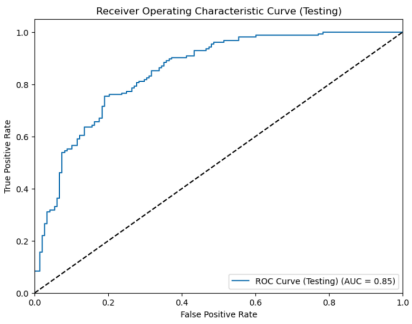
(C) ROC Curve for 12:00-12:29 vs 16:30-16:59



(D) ROC Curve for 14:30-14:59 vs 16:30-16:59



(E) ROC Curve for 15:00-15:29 vs 16:30-16:59



(F) ROC Curve for 16:00-16:29 vs 16:30-16:59

FIGURE 5.2: ROC Curve for different Time Windows

The ROC curves in Figure 5.2 reveal varying performance across different time intervals compared to the 16:30-16:59 reference period. Earlier intervals (Figures 5.2a,

5.2b, 5.2c and 5.2d) show nearly perfect discrimination, with curves close to the top-left corner indicating high accuracy. As the interval approaches 16:30-16:59 (plots 5.2e and 5.2f), curves shift towards the diagonal line, suggesting decreased classification performance likely due to similarity with the reference period for proximity.

In the next experiment, we will compare a baseline time period (excluding the start and closure times) with various alternative time intervals. Specifically, we will compare the time bucket from 11:00 A.M. to 11:29 A.M. with the following time windows: 9:00 A.M. to 9:29 A.M., 10:00 A.M. to 10:29 A.M., 11:30 A.M. to 11:59 A.M., 1:00 P.M. to 1:29 P.M., 3:00 P.M. to 3:29 P.M., and 4:30 P.M. to 4:59 P.M.. We can see the results in 5.2.

Window Comparison	KS Distance		AUC		Accuracy	
	Train	Test	Train	Test	Train	Test
9:00-9:29 vs 11:00-11:29	0.7330	0.7184	0.9343	0.9075	0.8665	0.8420
10:00-10:29 vs 11:00-11:29	0.5683	0.4515	0.8631	0.7967	0.7755	0.7086
11:30-11:59 vs 11:00-11:30	0.6238	0.5402	0.8906	0.8248	0.8061	0.7385
13:00-13:29 vs 11:00-11:30	0.5577	0.4753	0.8554	0.8059	0.7645	0.7283
15:00-15:29 vs 11:00-11:30	0.8412	0.8015	0.9705	0.9522	0.9022	0.8944
16:30-16:59 vs 11:00-11:30	0.8784	0.9010	0.9813	0.9801	0.9367	0.9410

TABLE 5.2: Comparison of Performance Metrics Across Several Time Windows vs 11:00-11:29

One noticeable observation is that the metrics measured during the time window 11:00-11:29 generally exhibit lower values than those presented in Table 5.1. This discrepancy occurs because this specific time window is less distinct compared to the closing time, which typically has more pronounced market activity or significance. Another observation is that as we move closer to the baseline time, 11:00-11:29, we observe consistent behaviors across adjacent time intervals. This suggests that market dynamics during these intervals exhibit similarities, possibly influenced by market trends or trading patterns.

Chapter 6

Application 3: AR Models as a Special Case of ES Models

This chapter is based on the article "Learning from the Past, Predicting the Statistics for the Future, Learning an Evolving System" (Levin, Lyons, and Ni, 2016). In this section, we will first establish a general framework for the Expected Signature Model. Following this, we will tailor this model to the context of time series analysis. To solidify our understanding, we will implement a computational example demonstrating that Autoregressive (AR) and Autoregressive Moving Average (ARMA) models can be viewed as specific instances of the Expected Signature Model.

6.1 The Expected Signature Model

First, let us define a Banach space. A Banach space is a complete normed vector space, meaning that it is a vector space equipped with a norm, and every Cauchy sequence in this space converges to a point within the space. This completeness property is crucial for various analytical techniques.

Consider two Banach spaces, V and W, and a compact time interval J. We can describe an E-valued data stream within the time interval J as a function $X : \mathcal{D} \to E$, where $\mathcal{D}$ represents the set of event times and $\mathcal{D} \subset J$.

To handle varying time stamps in the data streams, we embed X into a function that maps from J to E. One straightforward method is to extend X into a piecewise linear function as seen in 3.1.4, though there are more sophisticated methods available. For simplicity, we will assume that this embedded function is continuous and even piecewise smooth on a very fine scale. However, it may still exhibit high oscillations and be difficult to analyze directly.

The fine structure of these data streams is significant because it gives meaning to concepts such as iterated integrals and differential equations. Nevertheless, directly analyzing these streams in their raw form is neither convenient nor efficient. To address this, the theory of rough paths provides a solution by using a form of p-variation. This theory completes the space in such a way that integrals and differential equations become well-defined within this framework.

To clarify, $\mathcal{V}^p(J, E)$ denotes the space of continuous functions mapping from J to E with finite p-variation. In our discussion, we will consider elements of $\mathcal{V}^p(J, E)$ as representations of data streams or paths. This allows us to work within a robust

mathematical framework that supports effective analysis and application of regression techniques on data streams.

The effects of data streams can be modeled within a regression framework where the dependent variable is represented by observations on paths in $\mathcal{V}^p(J,E)$. In this context, the dependent variable itself may also be a stream. Let us consider observations of input-output pairs $\{X_i, Y_i\}_{i=1}^n$, where the relationship is assumed to adhere to:

$$Y_i = f(X_i) + \epsilon_i, \quad \forall i \in \{1,\dots,n\},$$

with $X_i \in V^p(J,E)$, $Y_i \in \mathcal{V}^p([0,T],W)$, $\mathbb{E}[\epsilon_i|X_i] = 0$, and f being an unknown function over the path space. The goal is to accurately and effectively estimating the function f within this framework.

Similar to classical nonparametric regression techniques in finite-dimensional cases, identifying specific feature sets within observed input-output data is crucial for linearizing the functional relationship between them. This chapter proposes applying linear regression on the signature features of a path to address this challenge. Throughout this chapter, we will adopt the following notation: $\mathbf{X}$ will represent the signature of X, and $\mathbf{Y}$ will represent the signature of Y.

Theorem 6.1.1 (Signature Approximation)**.** *Suppose $f : S_1 \to \mathbb{R}$ is a continuous function where S_1 is a compact subset of $S(V^p(J,E))$. Then for every $\epsilon > 0$, there exists a linear functional $L \in T((E))^*$ such that for every $a \in S_1$,*

$$|f(a) - L(a)| \leq \epsilon.$$

Theorem 6.1.1 states that for any small positive number ϵ, we can find a linear functional L such that for any point a in S_1, the difference between $f(a)$ and $L(a)$ is smaller than ϵ. One can see the proof of this theorem in Levin, Lyons, and Ni, 2016

Our objective is to learn the conditional distribution of Y given X. In the language of rough paths, this involves understanding the relationship between the expected values of Y based on the information from X. There are two main reasons to use signatures:

- Unique Path Determination: The signature of a path with limited changes (bounded variation) uniquely describes that path. This means the signature gives a full picture of the path's shape and movement.
- Expected Signature and Stochastic Processes: Under certain conditions, the expected signature of a stochastic process (random process) can determine the distribution of the random signatures. This means the average behavior of a process can be understood through its signature.

When the conditional expectation $\mathbb{E}[\mathbf{Y}|\mathbf{X}]$ is a continuous function of X, Theorem 6.1.1 tells us that this expectation can be approximated by a linear function of $\mathbf{X}$ in a local region. By incorporating a small amount of noise, we can derive a model known as the Expected Signature Model.

Definition 6.1.2. *The Expected Signature Model is used to describe the relationship between two stochastic processes X and Y, which take values in spaces E and W, respectively. The signatures of these processes, denoted by $\mathbf{X}$ and $\mathbf{Y}$, are assumed to be well-defined.*

The model claims that:

$$\boldsymbol{Y} = L(\boldsymbol{X}) + \epsilon,$$

where L is a linear functional that maps from the signature space of $\boldsymbol{X}$ to the signature space of $\boldsymbol{Y}$, and ϵ represents noise with an expected value of zero given $\boldsymbol{X}$ (i.e., $\mathbb{E}[\epsilon|\boldsymbol{X}] = 0$).

In essence, this model helps us understand and predict the behavior of Y based on the information encoded in the signatures of X.

By using these concepts, we can develop robust methods for analyzing and predicting complex data streams, leveraging the mathematical framework of rough paths and signature theory to simplify and solve regression problems in data science.

6.1.1 Calibration and Prediction

In the context of the expected signature model, when we have a large number of samples $\{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^N$, estimating the expected truncated signature of $\mathbf{Y}$ of order m, denoted as $\rho_m(\mathbb{E}[\mathbf{Y}|\mathbf{X}])$, essentially becomes a standard linear regression problem. Here, the coordinate iterated integrals of $\mathbf{Y}$ act as the multi-dimensional dependent variables, while the coordinate iterated integrals of $\mathbf{X}$ serve as the independent variables.

In practical applications, we focus on the truncated signature of $\mathbf{X}$ up to a certain order rather than the full signature, since we need a finite number of explanatory variables. Given that the regression is linear, we can utilize many existing linear regression techniques. To handle potential issues like collinearity in the design matrix and overfitting, we can apply regularization or variable selection methods such as LASSO or SVD. This calibration method is referred to as the ES approach, where ES stands for expected signature.

To evaluate how well the model fits, we use the mean squared error of the residuals $\{a_i\}_{i=1}^N$ as a performance metric, where:

$$a_i = \mathbf{Y}_i - \hat{f}(\mathbf{X}_i), \quad \forall i = 1, \ldots, N.$$

Alternatively, we can use R^2 or adjusted-R^2 as indicators of the model's fitting performance.

6.2 Expected Signature Model for Time Series

In this section, we explore the application of the expected signature model specifically to time series data. The expected signature model provides a powerful tool for analyzing and predicting the behavior of time series by leveraging the mathematical framework of signatures.

Consider a univariate time series $\{r_i\}_{i=1}^N$ and fix two positive integers p and q. For a given time index $k \in \mathbb{N}$, let $\mathcal{F}_k$ denote the information set available up to time t_k, comprising past returns before t_k. In this context, $\mathbf{Y}_k$ represents the signature of the future returns series $S(\{t_i, r_i\}_{k+1}^{k+q})$, and $\mathbf{X}_k$ represents the signature of the past returns series $S(\{t_i, r_i\}_{k-p}^{k})$.

We define the ES model as follows:

Definition 6.2.1. *Assuming the univariate time series $\{r_i\}_{i=1}^N$ is stationary, it is said to satisfy the **ES model** with parameters p, q, n, and m (denoted as $ES(p, q, n, m)$) if there exists a linear function $f : T^n(\mathbb{R}^2) \to T^m(\mathbb{R}^2)$ such that*

$$\rho_m(S(\{r_{t+i}\}_{i=1}^q)) = f(\rho_m(S(\{r_{t-i}\}_{i=0}^p))) + a_t,$$

where N is a positive integer such that $N \geq p + q$, and the residual terms a_t satisfy $\mathbb{E}[a_t|\mathcal{F}_t] = 0$.

Consider μ_k as the conditional expectation of the signature $S(\{t_i, r_i\}_{k+1}^{k+q})$, given the information up to time t_k. This is mathematically expressed as

$$\mu_k = \mathbb{E}[S(\{t_i, r_i\}_{k+1}^{k+q}) \mid F_k]. \tag{6.2.1}$$

Here, μ_k is represented as a function of X_k, indicating the existence of a mapping $f : T((\mathbb{R}^2)) \to T((\mathbb{R}^2))$ such that $\mu_k = f(\mathbf{X}_k)$.

It's important to note that μ_k and a_k belong to $T((E))$. The conditional covariance of the signature of the future return series $S(\{t_i, r_i\}_{k+1}^{k+q})$, given F_k, is defined by the function $\Sigma_k^2 : A^* \times A^* \to \mathbb{R}$:

$$\Sigma_k^2(I, J) = \mathrm{Cov}(\pi^I(\mathbf{Y}_k), \pi^J(\mathbf{Y}_k) \mid \mathcal{F}_k), \tag{6.2.2}$$

where $I, J \in A^*$.

The core assumption of the ES model rests on the stationarity of the time series $\{r_i\}$ (recall definition in Chapter 2), a standard requirement in time series analysis. This condition ensures that the distribution of the signature of $(r_{t_1}, \ldots, r_{t_k})$ remains invariant under time shifts.

In the $ES(p, q, n, m)$ model, it is assumed that the distribution of $r_{k+1}, \ldots, r_{k+q}$, given the current information $\mathcal{F}_k$, depends solely on the truncated signature (up to order n) of the p previous data points $r_{k-p}, \ldots, r_{k-1}, r_k$. These signatures are sufficiently rich to approximate any smooth mean function over the p preceding observations, in particular the signatures are enough to predict μ_k.

6.3 AR Models as a Type of Expected Signature Model

We have seen that autoregressive (AR) models are widely used in statistics and econometrics for modeling and estimating the conditional expectation of future values based on past information. Specifically, the AR model predicts the future value r_{k+1} based on a linear combination of past values $\{r_k, r_{k-1}, \ldots, r_{k-p+1}\}$, where p is the order of the model. In this section we are going to see that the AR model can be viewed as a special case of the Expected Signature (ES) model.

The AR model focuses on modeling and estimation of the conditional expectation m_k, and variance σ_k^2 of the future data r_{k+1} given the information up to time t_k, in other words:

$$m_k := \mathbf{E}[r_{k+1}|\mathcal{F}_k] \qquad \sigma_k^2 := \mathrm{Var}[r_{k+1}|\mathcal{F}_k]$$

Let μ_k be the expected signature of $\{(t_{k+i}, r_{k+i})\}_{i=0}^q$ on condition to $\mathcal{F}_k$ (where $q = 1$), so we can define the mean equation for r_t and the variance equation for r_t

using μ_k as follows:

$$m_k = \pi^{(2)}(\mu_k), \qquad \sigma_k^2 = 2\pi^{(2,2)}(\mu_k) - (\pi^{(2)}(\mu_k))^2,$$

where $\pi^{(2)}$ and $\pi^{(2,2)}$ are projections of the expected signature.

Since linear forms on the signature of p-lagged values of r_t are dense in the space of smooth functions on p-lagged values, the AR model aligns with the ES model framework. The AR model essentially uses the signature of past values as explanatory variables, thereby fitting into the ES model's structure.

6.4 Experiments and results

In this experiment, we aim to calibrate and predict time series data using two different approaches: the Autoregressive (AR) approach and the Expected Signature (ES) approach. We will compare the performance of these methods in terms of their predictive accuracy and ability to capture the underlying patterns of the time series data.

6.4.1 Data Generation

We begin by generating a synthetic AR series to serve as our dataset. The AR series is generated using the following AR(3) model parameters:

$$\phi = [0, 0.6, 0.15, -0.1]$$

where $\phi_0 = 0$ is the intercept, and $\phi_1 = 0.6$, $\phi_2 = 0.15$, $\phi_3 = -0.1$ are the coefficients for the lagged terms.

The AR series is generated with the function `generate_AR_series`, which includes random noise to simulate real-world data. The series length is set to $n = 4000$. Additionally, we generate an expected AR series without noise using the `generate_expected_AR_series` function for comparison.

6.4.2 AR Calibration

The Autoregressive (AR) approach is a linear model that predicts the current value of the series based on its past p values. The AR model is calibrated using linear regression. The data is split into a training set (80%) and a testing set (20%).

The model is trained to predict the next value in the series based on the previous $p = 3$ values. The calibration process involves fitting the following model:

$$r_t = \phi_0 + \phi_1 r_{t-1} + \phi_2 r_{t-2} + \phi_3 r_{t-3} + \epsilon_t$$

where ϵ_t is the random noise term.

6.4.3 ES Model Calibration

The Expected Signature (ES) model leverages the concept of signatures to capture the information in the time series. The signatures are computed using the `esig` library

from PyPI, 2024, which provides a way to encode higher-order interactions between data points.

The `ES_calibration` function applies the Expected Signature (ES) model to time series data with parameters p, q, n, m. Here, p denotes the number of past lags included in the past windows, while q represents the number of future values to be predicted. The integer n determines the order up to which signatures are computed for input data, and m specifies the level at which signatures are predicted for the output variables. The function operates on the univariate time series $\{r_i\}_{i=1}^{4}000$ generated, where each r_i represents a data point at time t_i. The goal is to model and predict future values using the ES approach.

The function iterates over segments of the training data to create past and future windows, guided by parameters p and q. For each time index t, it constructs:

- Past path: The past path encapsulates historical data up to time t. Specifically, the past path $\{r_i\}_{i=t-p}^{t}$ comprises data points ranging from r_{t-p} to r_t, covering $p+1$ data points. Additionally, for each past path at time t, an initial point with a value of 0 is included at time $t-p-1$.
- Future Path: This projects expected future values based on the data at t. The future path $\{r_i\}_{i=0}^{t+q}$ forecasts data points from r_t to r_{t+q}, predicting $1+q$ data points ahead (the present point plus q future data points). Also, we add an additional initial point at time $t-1$ with zero value.

For each time index t, the function computes two signatures. $\rho_n(S(\{r_i\}_{i=t-p-1}^{t}))$ for the past path, denoted as X_k, which serves as independent variables and it also computes $\rho_m(S(\{r_i\}_{i=t-1}^{q}))$ for the future path Y_k, representing dependent variables to be predicted using the information provided by X_k. Here, ρ_m represents the signature up to a specified truncation order m, capturing the key features of the path in a concise manner. We use the model $ES(3,1,4,2)$. The choice of 3 for the first parameter is because we are predicting an AR(3) model, which relies on three lagged values. The second parameter is set to 1 since we are only predicting one future value. The third parameter, 4, is selected as the signature level for the input data, which is sufficient to predict the output effectively. Lastly, the fourth parameter is 2, chosen because signatures up to order 2 capture the most important features necessary for prediction.

The resulting signatures `X_sig` and `y_sig` are stored as arrays. The data is then split into training and testing sets using an 80-20 split ratio, where the training set is used to fit the predictive model, and the testing set is used to evaluate its performance.

A Lasso model is trained using the training data `X_sig_train` and `y_sig_train` with `alpha` $= 0.01$. This model learns the relationship between past signatures X_k and future signatures Y_k, aiming to predict future values based on historical patterns captured by the signatures. As demonstrated in 3.1.13, we focus exclusively on the third coordinate of the signature, which corresponds to the value of r_t that we are attempting to predict.

6.4.4 Performance Analysis

The performance of both approaches are evaluated using the following metrics:

- R^2: Coefficient of determination.
- Adjusted R^2: Adjusted for the number of predictors.
- MSE: Mean Squared Error.
- Running time: The total time taken to execute the model.

6.4.5 Results and Comparison

The performance metrics for both the AR and ES approaches are computed and compared. Note that we can compute the metrics against the real values (with noise) and the expected values, the results are presented in Table 6.1 and in Table 6.2, respectively.

Approach	R^2	Adjusted R^2	MSE	Running time
AR Approach	0.39	0.36	1.0633	0.0 s
ES Approach	0.39	0.39	1.0664	1.5 s

TABLE 6.1: Performance metrics for AR and ES models against real values (with noise).

Approach	R^2	Adjusted R^2	MSE	Running time
AR Approach	0.99	0.99	0.0008	0.0 s
ES Approach	0.99	0.99	0.0033	1.59 s

TABLE 6.2: Performance metrics for AR and ES models against expected values.

Table 6.1 shows the performance metrics against real values with noise. Both the AR and ES approaches exhibit identical R^2 and Adjusted R^2 values of 0.39, indicating that they explain approximately 39% of the variance in the data when noise is present. The AR model achieves a slightly lower MSE of 1.0633 compared to the ES model's MSE of 1.0661. Notably, the AR model demonstrates instantaneous computational time (0.0 s), whereas the ES model requires 1.5 seconds to complete computations.

Table 6.2 evaluates performance against expected values, where both models show exceptional results with R^2 and Adjusted R^2 values of 0.99, indicating they explain 99% of the variance in the data against the expected values. The AR model achieves an impressively low MSE of 0.0008, while the ES model exhibits a slightly higher MSE of 0.0037. Running time results are exactly as in Table 6.1.

These results underscore that the AR model can be viewed as a subset or a particular case within the broader framework of the ES model. The AR model's ability to fit closely to expected values with minimal MSE and instantaneous computational time further supports its effectiveness. Meanwhile, the ES model, while achieving comparable R^2 values, mathematically demonstrates the capacity to handle more complex data representations through its signature-based approach, but with slightly higher computational requirements.

In addition to the performance statistics, visualizing the results can provide a deeper understanding of how each approach performs. To this end, we have created

several plots to compare the predicted and actual values for both the AR and ES approaches.

First, we plot the predicted time series against the expected time series for the test set. Figures 6.1 and 6.2 show these plots for the AR and ES approaches, respectively. For both approaches, we can see an almost perfect overlap between the predicted and expected data, suggesting that both models predict well the expected data of the test test.

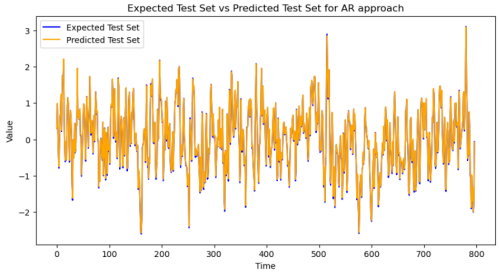


FIGURE 6.1: AR Approach: Expected Values vs. Predicted Values

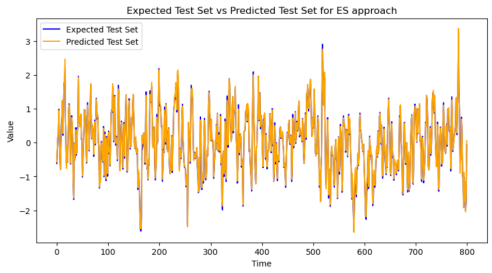


FIGURE 6.2: ES Approach: Expected Values vs. Predicted Values

Next, we examine the mean of the true series versus the mean of the predicted series. Figures 6.3 and 6.4 display these comparisons for the AR and ES approaches, respectively. In these figures, we plot the true mean of the series against the predicted mean. For both approaches, the predicted values lie almost perfectly along the regression line, confirming the models' accuracy.

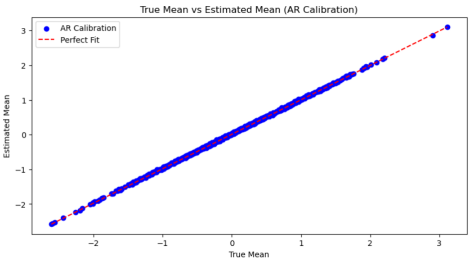


FIGURE 6.3: AR Approach: True Mean vs. Predicted Mean

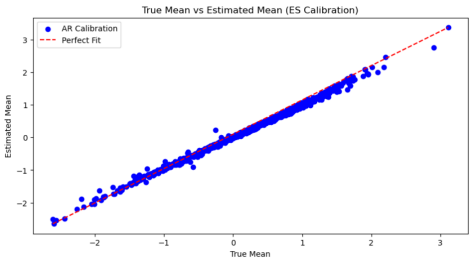


FIGURE 6.4: ES Approach: True Mean vs. Predicted Mean

These visualizations reinforce the conclusions drawn from the performance metrics. The AR model aligns closely with the capabilities of ES models. This suggests that AR models can be viewed as a specific instance within the broader framework of ES models, leveraging signature-based approaches to achieve reliable predictions in time series analysis. Also, this experiment demonstrates the effectiveness of the ES model in predicting time series data, in particular AR models.

Chapter 7

Conclusions

This thesis has delved into the versatile applications of the signature method within the realms of time series analysis and financial data streams, showcasing its efficacy in both theoretical exploration and practical implementation.

Our journey began with a rigorous exploration of the theoretical base of the signature method, emphasizing its role in preserving essential dynamics while simplifying complex temporal information. Subsequently, through three distinct applied studies, we demonstrated its effectiveness across different domains.

In the first application we conducted a classification analysis of Autoregressive (AR), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA) models using signature-based features. This study highlighted the method's ability to distinguish between various model structures based on their signature characteristics.

In the second application we used signature terms as features for classifying crude oil data streams based on 30-minute intervals demonstrating the method's practical utility in temporal segmentation tasks. By accurately capturing temporal patterns, the signature method facilitated meaningful segmentation and analysis of time-sensitive data.

In the last application we investigated the relationship between AR models and Expected Signature (ES) models providing insights into how AR models can be interpreted within the broader ES framework. This exploration emphasized the method's versatility in accommodating different model structures while maintaining predictive accuracy.

Each study underscored the signature method's capability to transform complex temporal data into interpretable and actionable insights. By integrating theoretical insights with practical applications, this thesis has successfully achieved its objectives.

Bibliography

Akivis, M.A., V.V. Goldberg, and R.A. Silverman (1972). *An Introduction to Linear Algebra and Tensors*. Dover Books on Mathematics Series. Dover Publications. ISBN: 9780486635453. URL: https://books.google.es/books?id=0SvH-j5qr6wC.

Brockwell, Peter J. and Richard A. Davis (2016). *Introduction to Time Series and Forecasting*. Springer Texts in Statistics. Springer International Publishing. DOI: 10.1007/978-3-319-29854-2. URL: https://doi.org/10.1007/978-3-319-29854-2.

Chevyrev, Ilya and Andrey Kormilitzin (2016). *A Primer on the Signature Method in Machine Learning*. arXiv: 1603.03788 [stat.ML]. URL: https://arxiv.org/abs/1603.03788.

Gyurkó, Lajos Gergely et al. (2014). *Extracting information from the signature of a financial data stream*. arXiv: 1307.7244 [q-fin.ST]. URL: https://arxiv.org/abs/1307.7244.

Hambly, Ben and Terry Lyons (Mar. 2010). "Uniqueness for the signature of a path of bounded variation and the reduced path group". In: *Annals of Mathematics* 171.1, 109–167. ISSN: 0003-486X. DOI: 10.4007/annals.2010.171.109. URL: http://dx.doi.org/10.4007/annals.2010.171.109.

HistData.com (2024). *HistData: Historical Data for Forex Trading*. URL: https://www.histdata.com/.

Levin, Daniel, Terry Lyons, and Hao Ni (2016). *Learning from the past, predicting the statistics for the future, learning an evolving system*. arXiv: 1309.0260 [q-fin.ST]. URL: https://arxiv.org/abs/1309.0260.

PyPI (2024). *esig: Library for computing signature features*. https://pypi.org/project/esig/.

Remy, Philippe (2024). *FX-1-Minute-Data*. URL: https://github.com/philipperemy/FX-1-Minute-Data.