The School of Mathematics

THE UNIVERSITY
of EDINBURGH

Machine Learning and Rough Path

by

Zihao Song

Dissertation Presented for the Degree of MSc in Computational Mathematical Finance

August 2019

Academic Supervisor: Dr. Yvain Bruned

Machine learning and rough path

Zihao Song

August 2019

Contents

Abstract

The signature method is one of the most prevalent tools in machine learning, especially in classifying and character recognizing, achieving lots of brilliant application. This essay is based on the theoretical expression of the signature method, the way it connect with solution of differential equation driven by rough path, the application of it in classification using Python, and, to be extended more on theoretical, the rooted tree as well as its relation with B-series representation solution of differential equation driven by branched rough path. It seems that, essentially, we discuss around the series solution of a differential equation more and more deeply, with the use of the signature method and rooted tree. This a new methodology for machine learning to recognize the path and we do believe it will form another way to get features on a path.
Keywords: Signature method; machine learning; rough path; rooted tree; series solution; differential equation

1 History and motivation

This section is about the history and motivation of the signature method of the rough path in machine learning.

1.1 Machine learning

Machine learning was formed by Arthur Samuel, a pioneer in American computer games and artificial intelligence, and he worked at IBM in 1959. From that time, it becomes an advanced scientific endeavour, and it grew out of the quest for artificial intelligence. Until now we have founded several algorithms including (Un)Supervised learning, Reinforcement learning, Feature learning, Sparse dictionary learning, Anomaly detection as well as Association rules [ML19].
When it began to prosper, ML, reorganized into an independent field in the 1990s and began to be known to everyone. Then scientists have changed the goal of solving practical problem-solving problems. We turn our attention from the symbolic approach of artificial intelligence inheritance to the methods and models obtained through statistics and probability theory. It is increasingly benefiting from the growing popularity of digital information and the ability to distribute information over the Internet. People have made a lot of achievements in machine learning, and we will have more in the future. It is not a field of science, but a way of life, a way of thinking [ML19].

1.2 Signature method of Rough path

Rough paths, captures and interactions between precise high-oscillation and non-linear systems have long been a field of science. The founders of its technology are LC Young and KT Chen. The concept and theorem, as well as its unified estimation, have been widely used. Currently, it is used (Graham) to automatically recognize Chinese handwriting and (Hairer) to develop appropriate SPDEs to simulate a stochastic evolution interface [Lyo14].
The signature of a path is a character that is the homomorphism of a group element from the monoid of paths to a closed tensor algebra. It provides a "conclusion" of the path. Recalling its history, Hambly and Lyons have demonstrated that this non-exchange transformation is accurate for paths with finite variation conditions until the appropriate zero modification. The "conclusions" of these gradations or the characteristics of the path are the core of the rough path, in fact, all of which require attention to the entire structure of the path. Taylor's theorem states how any smoothing function is treated locally as a linear combination of certain functions [Lyo14].
Then in rough path theory, the signature was studied by Chen for the first time with the piecewise smooth paths. Currently, the path signature has been used by Lyons to search

for the solution of differential equations driven by rough signals. Its condition has been extended by Lyons from bounded variation to finite p-variation for any p $\geqslant$1[YLNSJC17]. Nowadays, the rough path theory is developing to further more complex situation and it would achieve more result in applications.

2 Review of Signature method of Rough path

As developing during a long time, the signature method has formed a series of theoretical properties and has recently gained attention in the mathematical community in part due to its connection with Lyons' theory of rough paths. Thus, in this part, we review the formal theories and give own understanding.

2.1 Path and its integral

Definition 1. *In mathematics, a path in a topological space X is a continuous function f from the measurement interval $I = [a,b] \to X$.[CK16]*

We will use the normal notation $X_t = X(t)$ to denote the parameter $t \in [a, b]$ dependently. Moreover, we need to check whether is a path differential or not and to fit the real-world path or to form an easier model, we always think that paths are piecewise differentiable. For a regular smooth path, it has derivatives of all orders.
This parametrisation generalizes in d-dimensions $X_t \in \mathbb{R}^d$ as:

$$X_t = \{X_t^1, X_t^2, X_t^3, ..., X_t^d\}. \tag{2.1}$$

Let us consider a simple piecewise linear path as an example in $\mathbb{R}^2$ is:

$$Y_t = \{Y_t^1, Y_t^2\} = \{t, f(t)\}, t \in [0, 1]. \tag{2.2}$$

As for the normal integrals of the path, we always consider the Riemann integral of a real-valued function with a condition like continuous and bounded as below:

Definition 2. *For a one-dimensional path $X : [a, b] \mapsto \mathbb{R}$ and a function $f : \mathbb{R} \mapsto \mathbb{R}$, the path integral of X against f is:*

$$\int_a^b f(X_t)dX_t = \int_a^b f(X_t)\dot{X}_t dt, \tag{2.3}$$

in which $\dot{X}_t = dX_t/dt$ is called the "upper-dot".

Remark 1. *When we think more about the signature method of a rough path, we need to know exactly the Riemann-Stieltjes integral of one path and the other, and it is similarly as above: for any path $Z : [a, b] \mapsto \mathbb{R}$ against a path $Y : [a, b] \mapsto \mathbb{R}$, we have:*

$$\int_a^b Z_t dY_t = \int_a^b Z_t \dot{Y}_t dt. \tag{2.4}$$

Considering why we need to research about the path, one key point is they represent the evolution hidden in some wider system which is interacted and influenced by that process. Another is that for most paths, it is sure for is to find the content feature in standard presentations because they are locked inside its complex multidimensional oscillation. So, it is necessary and equivalent to understand the trend and use it to direct our future activity.

2.2 Signature

2.2.1 Build the definition of signature

To lead out the definition of signature, we need to state the tensor algebra space as below, in which the signature of a path takes values [Lyo07].

Definition 3. *A tensor is a multi-linear function that can be used to represent a linear relationship between some vectors, scalars, and other tensors, with a form like a multiple linear mapping on a vector space or its dual space.*
A sequence of tensors formed a formal E-tensor series which we write as $a = (a_0, a_1, ...)$*, where* $a_n \in E^{\otimes n}$*.*

In order to form an algebra, we introduce two binary operations on E-tensor series, "$+$" addition and "$\otimes$" product, with the definition from [Lyo07]

Definition 4. *Let* $x = (x_0, x_1, ...)$*,* $y = (y_0, y_1, ...)$ *be two E-tensor series, we have:*

$$x + y = (x_0 + y_0, x_1 + y_1, ...); \tag{2.5}$$

and

$$x \otimes y = (z_0, z_1, ...), \tag{2.6}$$

where for each $n \geqslant 0$*,*

$$z_n = \sum_{k=0}^{n} x_k \otimes y_{n-k}. \tag{2.7}$$

To have a complete notation of tensor product, we need to rewrite $a \otimes b$ as ab, and add 1 which is for series $(1, 0, ...)$, 0 for series $(0, 0, ...)$ as well. If $\lambda \in \mathbb{R}$, then we define $\lambda a = (\lambda a_0, \lambda a_1, ...)$.

Remark 2. *Then the tensor algebraic space* $T((E))$ *is defined as the vector space of all formal E-tensor sequences.*

Then we state the definition of the signature using the tensor algebra.

Definition 5. *The signature of a path* X*:* $[a, b] \mapsto \mathbb{R}$*, denoted by* $S(X)_{a,b}$*, can be simply regarded as a formal infinite sum of non-commutative tensor products, and the coefficient of each part is from its coordinate iterated integral corresponding to each respective order, which is in the form:*

$$S(X)_{a,b} = (1, S(X)^1_{a,b}, S(X)^2_{a,b}, ..., S(X)^d_{a,b}, S(X)^{1,1}_{a,b}, S(X)^{1,2}_{a,b}, ...), \tag{2.8}$$

where the superscripts run with all multi-indexed collections

$$W = \{(i_1, ..., i_k) | k \geqslant 1, i_1, ..., i_k \in \{1, ..., d\}\},$$

and for every multi-index $I = (i_1, ..., i_n)$*, we have*

$$S(X)^I_{a,b} = \int_{a<t_k<b} \cdots \int_{a<t_1<t_2} dX^{i_1}_{t_1} \otimes \cdots \otimes dX^{i_k}_{t_k}, \tag{2.9}$$

which could be simply write in normal $\mathbb{R}^d$ *field as real-valued path as:*

$$S(X)^I_{a,b} = \int_{a<t_k<b} \cdots \int_{a<t_1<t_2} dX^{i_1}_{t_1} \cdots dX^{i_k}_{t_k}. \tag{2.10}$$

Example 1. *For example, let us consider the special cases of first two levels signature:*

$$S(X)^i_{a,b} = \int_{a<s<b} dX^i_s. \tag{2.11}$$

$$S(X)^{i,j}_{c,d} = \int_{c<s<d} S(X)^i_{c,s} dX^j_s = \int_{c<r<s<d} dX^i_r dX^j_s. \tag{2.12}$$

For the essence of the signature of one path, as we discuss briefly before, it is exactly the information about a path which you could use to predict how the output of the system will behave in the future, along with using the generalization of Taylor's theorem. Thus, if we need to state the relation of this method with machine learning, it is natural that it is the right information to use for a general machine-learning algorithm to understand the shape of the path. Because the length of the set of signature depends on the numbers of levels, so we may not able to store the whole signature of a path on a computer. But in fact, or naturally, when we calculate more levels of signature, the shape of the path is determined more precisely. As we experience on the computer using Python package iisignature, if we use a path changes very slightly, it seems the first few levels of its signature has the same changing trend. If a path retains its shape no matter its value change or not, its signature will not change[RG18].

2.2.2 The first/second level of signature and its geometric intuition

The first level of signature of the path, as mentioned above, is the collection of d numbers in real field

$$S(X)^1_{a,b},\ldots,\ S(X)^d_{a,b},$$

which is simply the increment of the path X, and the second level is the collection of d^2 numbers in real field

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

which have different geometric intuition for different form. For $S(X)^{i,i}_{a,b}$, we note that it is equal to $(X^i_b - X^i_a)^2/2$, which is a special case of the shuffle product which we shall review in next part, and for the $S(X)^{i,j}_{a,b}$ part, consider the *Lévy* area:

Definition 6. *As shown in figure, the Lévy area is what we called as a signed area enclosed by the edge of path and the line connected two endpoints, and we calculate it as the sum of the total areas of the regions A_+ and A_-, which is the area times the frequency of the path goes round that region anticlockwise minus the frequency of the path goes round it clockwise[RG18], like for 2 dimensional path $\{X^1_t,\ X^2_t\}$, the Lévy area is*

$$A = \frac{1}{2}(S(X)^{1,2}_{a,b} - S(X)^{2,1}_{a,b}). \tag{2.13}$$

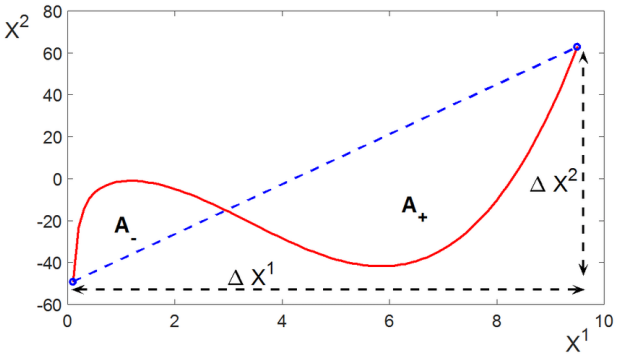

Figure 1: Example of signed Levy area [CK16]

2.2.3 Use Picard iteration and the Signature to compute the solution of the controlled differential equation

As we have known, most differential equations could have a proof for the existence and uniqueness of the solution to a first-order differential equation. For higher-level solution, always we could use the method of successive approximations. Computers can do all the boring super-position algorithm thus we can get a better grasp of solution.
For its history, there is a technique for stating the iteration and proving that a solution exists, which goes back to Emile Picard. Along with the history, the iteration method has been used in proofing the existence and calculating approximate solutions of a differential equation for at least 180 years before nowadays.
Liouville was acclaimed for figuring out this method in the 1830s and the application to the context of linear boundary-valued differential equations such as one-dimensional heat conduction problems. Furthermore, as mentioned, Picard makes a significant contribution to the development of "Picard iterations" for ODE or PDE systems with initial values and boundary values. In the 20th century, considerable growth occurred in theory and application. For example, Gau extended the idea of using Picard in iteration to the case of integer order $n > 2$. In recent year, Diethelm and Ford applied the method of iterations to the case for any order of integer is greater than zero as well as fractional order (2002) [C19].
Let us firstly state the classical Picard-Lindelöf theorem[CK16]

Theorem 1. *Let us consider the Linear Controlled Differential Equations:*

$$\frac{dy(t)}{dx(t)} = Ay(t), y_0 = a, \tag{2.14}$$

Where the control path $x(t)$ is fixed and smooth, called it as a simple rough path, and this is an example of a differential equation driven by rough path, all the state space and vector fields are linear, then for y, as what we figure about the solution of differential equation, is a value varies around the starting location a linearly. And also Suppose F is a Banach space, and A is a linear map $E \to Hom_R(F, F)$ and that $x(t)$ is a control path in E.
Then we would use Picard's method to compute an approximated solution to it in an iterative sequences form and connect it with the signature method. The integral form of it is given by

$$y(t) = y(0) + \int_0^t Ay(s)dx(s). \tag{2.15}$$

Then we could define a sequence of functions $y_k(t), k = 0, 1, ...,$ where the first term is $y_0(t) = y(0)$, and for $k \geqslant 1$, we define

$$y_k(t) = y(0) + \int_0^t Ay_{k-1}(s)dx(s). \tag{2.16}$$

Then, under suitable conditions, the solutiont is given by

$$y(t) = lim_{k\to\infty} y_k(t). \tag{2.17}$$

Thus, we use this idea and the signature to think about computing the solution of this controlled differential equation:
For all $t \in [a, b]$ as below, use I_e to denote the identity operator (or matrix) in the vector space of $e \times e$ real matrices:

$$y_0(t) = y(0) = a; \tag{2.18}$$

$$y_1(t) = y(0) + \int_0^t Ay_0(s)dx(s) = (A\int_0^t dx(s)) + I_e)(y(0)); \tag{2.19}$$

$$y_2(t) = y(0) + \int_0^t Ay_1(s)dx(s) = (A^2 \int_0^t \int_0^s dx(u)dx(s) + A\int_0^t dx(s) + I_e)(y(0)); \quad (2.20)$$

$$\vdots$$

$$y_n(t) = y(0) + \int_0^t y_{n-1}(s)dx(s) = (\sum_{n=1}^{\infty} A^n \int_{0<t_1<\ldots<t_n<t} dx(t_1)\cdots dx(t_n) + I_e)(y(0)). \quad (2.21)$$

$$\vdots$$

One can check, the final term of n-th equation is completely determined linearly by the k-th level of the signature $S(X)_{a,t}$ of X at time $t \in [a,b]$.
This is still more about the convergence about this n-th iteration, like this remark:

Remark 3. *[Lyo07]If $x(t)$ is a path finite variation on I: $[0,t]$ with length $|x_I| < \infty$, then we find the final term of n-th equation is*

$$S_I^n := \int_{0<t_1<\ldots<t_n<t} dx(t_1)\cdots dx(t_n) \leq \frac{|x_I|^n}{n!}. \quad (2.22)$$

giving uniform error control:

$$\|y(t) - \sum_{n=0}^{N-1} A^n \int_{0<t_1<\ldots<t_n<t} dx(t_1)\cdots dx(t_n)y(0)\| \leq (\sum_{n=N}^{\infty} \frac{\|A\|^n|x_I|^n}{n!})\|y(0)\|. \quad (2.23)$$

This is the first type of Picard iteration computing to generate a "sequence of numbers" which converges to a solution of a "good" differential equation. Besides that, we could also illustrate the second type use on efficiently generating a "sequence of functions" which converges to a solution of an ODE or PDE, which is more complex and we would not mention here.

2.3 Properties of the general signature

In this section, let us review several fundamental properties of the signature. All will not provide the proof details of these properties, and we emphasize that it is easy to figure out they are all straight-forward consequences of classical integration theory. The properties of the signature show the operational law between different items and provide the normal algebraic structure of signature, which could make the path operation more theoretical.

2.3.1 No-change under time reparametrisations

Definition 7. *We consider the reparametrization as a continuous, no reduction, surjective function $\psi : [a,b] \to [a,b]$, then we can see: Let X, Y: $[a,b] \to \mathbb{R}$ be two paths in real field and $\psi : [a,b] \to [a,b]$ be reparametrization. Define the paths $\tilde{X}$ and $\tilde{Y}$: $[a,b] \to \mathbb{R}$ by*

$$\begin{aligned} \tilde{X}_t &= X_{\psi(t)}, \\ \tilde{Y}_t &= Y_{\psi(t)}. \end{aligned} \quad (2.24)$$

Then, we obtain:

$$\dot{\tilde{X}} = \dot{X}_{\psi(t)}\dot{\psi}(t). \quad (2.25)$$

Proposition 1. *From which we could figure out that:*

$$\int_a^b \tilde{Y}_t d\tilde{X}_t = \int_a^b Y_{\psi(t)}\dot{X}_{\psi(t)}\dot{\psi}(t)dt = \int_a^b Y_{\psi(t)}dX_{\psi(t)}. \quad (2.26)$$

This shows that path integrals are equal under a time reparametrization of both paths, then, the signature $S(X)_{a,b}$ remains invariant under time reparametrizations of path X. In real world, if we say a system is time-invariant then with an arbitrary delay, it commutes with the same delay. In our signature world, it seems that this means if a path changes parameter but has the same shape, those two paths would have the same signature.

2.3.2 Character property

Now, we could operate the signature as an element called character, and actually we focus on how the indexes change.
The shuffle product was introduced in 1953. The name tends to mean to the fact that this operator can be regarded as the overall ways of riffle shuffling two words together: this is such a kind of permutation. Also, this product is similar to the simple product: commutative and associative[EML53].
Looking back through the founding article [EML53], they state the shuffle product by reviewing the Cartesian product and then defined it on the Cartesian product of any two FD-complexes. For a (p, q) shuffle, it is defined to be a respective partition of the set $[p + q - 1]$ of integers result as two disjoint sets of p and q integers. The meaning of a "shuffling" way is we have a set of p cards and one of q cards, we shuffle them together but with themselves both taking the remaining positions when we check any one of these two sets. A shuffle can also be described as a permutation as we discuss below:
Let us essentially define the shuffle product of two multi-indexes, in order to understand the product of two signatures which are called the character of the path.

Definition 8. *Let us consider the structure of the character, which is a shuffle algebra (a Hopf algebra) with a basis which corresponding to words on some multi-indexes. We could simply define the product and the sum of this algebra with its empty element ϵ and generator element i as:*
For the sum of multi-indexes, we check the definition of (k, m)-shuffles: it is the subset of permutation in the symmetric group S_{m+n} defined by two multi-indexes $I = (i_1, ..., i_k)$ and $J = (j_1, ..., j_m)$ with all items from $\{1, ..., d\}$. Define the operator $I + J$ as a multi-index:

$$(r_1, ..., r_k, r_{k+1}..., r_{k+m}) = (i_1, ..., i_k, j_1, ..., j_m). \tag{2.27}$$

Then, we use a simple way to define the Shuffle product inductively by:

$$u \sqcup\!\sqcup \epsilon = \epsilon \sqcup\!\sqcup u = u; \tag{2.28}$$

$$au \sqcup\!\sqcup bw = a(u \sqcup\!\sqcup bw) + b(au \sqcup\!\sqcup w), \tag{2.29}$$

where the ϵ is the empty word, a and b are single elements, and u and w are arbitrary multi-indexes.
Thus, we get an algebra of character with shuffle product and sum.

From above, we could get the product rule of two signatures:

Proposition 2. *Consider $X : [a, b] \to \mathbb{R}^d$ as a path in real field and $I = (i_1, ..., i_k)$ and $J = (j_1, ..., j_m)$ as two multi-indexes with all items from $\{1, ..., d\}$, it holds that*

$$S(X)^I_{a,b} S(X)^J_{a,b} = \sum_{K \in I \sqcup\!\sqcup J} S(X)^K_{a,b}. \tag{2.30}$$

Example 2. *Let us consider a simple example of a path $X : [a, b] \to \mathbb{R}^2$ which is two dimensional.*

The shuffle product implies that:

$$S(X)^{i,j}_{a,b}S(X)^{u,v}_{a,b} = S(X)^{u,v,i,j}_{a,b} + S(X)^{u,i,v,j}_{a,b} + S(X)^{u,i,j,v}_{a,b} + S(X)^{i,u,v,j}_{a,b} + S(X)^{i,u,j,v}_{a,b} + S(X)^{i,j,u,v}_{a,b}. \tag{2.31}$$

2.3.3 Chen's identity

We then need to discuss the algebraic relationship between paths and their signatures, which is named as Chen's identity. It asserts if one path is the concatenation of two paths, its signature is the tensor product of the signature of each path[Ni14].
Research back to the founding article of Chen's identity [Chen58], we notice that it would be clear to state identity by expressing how to define the integral abnormally. For any two paths A and B in $\mathbb{R}$, Chen checked:

$$\int_{A*B} \pi_1 \cdots \pi_n = \int_B \pi_1 \cdots \pi_n + \cdots + \int_A \pi_1 \cdots \pi_i \int_B \pi_{i+1} \cdots \pi_n + \cdots + \int_A \pi_1 \cdots \pi_n \tag{2.32}$$

$$\int_{A*A^{-1}} \pi_1 \cdots \pi_n = 0 \tag{2.33}$$

where $\pi_1 \cdots \pi_n$ are differentials in a differentiable manifold $\mathbb{R}$ and A^{-1} is the space where the path in A undergoes a change of parameter.
Then he defined for $\pi_1 \cdots \pi_n$ as above and $X_1, \cdots, X_n$ distinct non-commutative indeterminate, with an exponential homomorphism, or to be more related with our essay, the power series

$$\Phi(A) = 1 + \sum_{n=1}^{\infty} \sum \int_A \pi_{i_1} \cdots \pi_{i_n} X_{i_1} \cdots X_{i_n}, \tag{2.34}$$

and he stated that $\Phi(A * B) = \Phi(A)\Phi(B)$, which we call "Chen's identity" currently.
We firstly rewrite the signature in form of formal power series in d indeterminates. Suppose that $e_1, ..., e_d$ be a basis of E, and thus for every $k \geqslant 0, e_{j_1}, ..., e_{j_k}$ is a basis of $E^{\otimes n}$, then the signature can be written as:

$$S(X)_{a,b} = 1 + \sum_{k=1}^{\infty} \sum_{j_1,\ldots,j_k \in \{1,\ldots,d\}} S(X)^{j_1,\ldots,j_k}_{a,b} e_{j_1} \otimes \cdots \otimes e_{j_k}. \tag{2.35}$$

Before stating the Chen's identity, it remains to define the concatenations of paths in [CK16] as:

Definition 9. *For the path $X : [a, b] \to \mathbb{R}^d$ and the path $Y : [b, c] \to \mathbb{R}^d$, we define their concatenation as the path $X * Y : [a, c] \to \mathbb{R}^d$ as:*

$$(X * Y)_t = \begin{cases} X_t & t \in [a, b]; \\ X_b + (Y_t - Y_b) & t \in [b, c]. \end{cases}$$

Then we could state the Chen's identity, which shows the signature turns the concatenation product into the product $\otimes$:

Proposition 3. *Let $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$ be two simple paths. Then*

$$S(X * Y)_{a,c} = S(X)_{a,b} \otimes S(Y)_{b,c}. \tag{2.36}$$

2.3.4 Time-reversal

Reason for discussing the time-reversal is it states that the signature of a path precisely stay inverse under the product $\otimes$ with the signature of that path running X backwards

in time [CK16].
Firstly let us define the notation of time-reversal:

Definition 10. *for path X, we define the time-reversal of it as the path $\overleftarrow{X} : [a,b] \to \mathbb{R}^d$ and it holds the position $\overleftarrow{X}_t = X_{a+b-t}$ for all $t \in [a,b]$.*

Then we get property of the signature for Time-reversed path: For a path X, we can figure out that:

$$S(X)_{a,b} \otimes S(\overleftarrow{X})_{a,b} = 1. \tag{2.37}$$

We can find the proof of this equation from the position 2.14 of [CK16]. In this equation, we need to explain the element 1, which is in the algebra of general power series that is not commuting in d-indeterminates as the form:

$$\sum_{k=0}^{\infty} \sum_{i_1,...,i_k \in \{1,...,d\}} \lambda_{i_1,...,i_k} e_{i_1} \otimes \cdots \otimes e_{i_k}, \tag{2.38}$$

where the second summation runs over all multi-indexes $\{i_1, ..., i_k\} \in \{1, ..., d\}$, and $\lambda_{i_1,...,i_k}$ are numbers in real field.
Thus, it is the element when $\lambda_{i_1,...,i_k} = 0$ and $\lambda_0 = 1$ for all $k \geqslant 1$ and $i_1, ..., i_k \in \{1, ..., d\}$.

2.3.5 Log signature

Signature of a path, as Terry said in [Lyo14], is actually taking its values in a curved subspace of the tensor algebra, and it is a homomorphic concatenation of path segments into the algebra, producing the inverse tensor by reversing the path segments. Thus, there are a range of map about signature as a curved space in the tensor series and one important map for signature is the logarithm, and it is one to one on the group and provides a flat parametrization of the group in terms of elements of the free power series.
The log signature is a compressed version of the signature. It carries the same information, but much more compact. As we say the signature is a power series, So let's first discuss the logarithm of the power series algebra in the usual sense [RG18].

Definition 11. *For a power series:*

$$x = \sum_{k=0}^{\infty} \sum_{j_1,...,j_k \in \{1,...,d\}} \lambda_{j_1,...,j_k} e_{j_1} \otimes \cdots \otimes e_{j_k}, \tag{2.39}$$

which $\lambda_0 > 0$, we have its log is:

$$logx = log(\lambda_0) + \sum_{n \geqslant 1} \frac{(-1)^n}{n} (1 - \frac{x}{\lambda_0})^{\otimes n}, \tag{2.40}$$

where the $\otimes n$ is the n-th power in the tensor algebra.

Example 3. *For a real number $\lambda \in \mathbb{R}$ and the series*

$$x = 1 + \sum_{k \geqslant 1} \frac{\lambda^k}{k!} e_1^{\otimes k}, \tag{2.41}$$

we can easily figure out that its log is:

$$logx = \lambda e_1. \tag{2.42}$$

Thus, we use the same notation as in the real-algebra, $logS(X)_{a,b}$, to express the log signature. Because under $\otimes$ and $+$, the space $T((E)) := \bigoplus_0^\infty E^{\otimes n}$ is a unital associative algebra, which means it is also a Lie algebra [Lyo14], and with Lie-bracket as below:

Definition 12. *For two formal power series x and y, we have:*

$$[x, y] = x \otimes y - y \otimes x. \tag{2.43}$$

Also we define several notations and from those we could figure out the structure of log signature.
We use the notation $\mathcal{L}(E)$ for the algebra generated by E which is the space of Lie polynomials, $\mathcal{L}^{(n)}(E) : \mathcal{L}(E) \to T^{(n)}(E) := T((E))/\bigoplus_{n+1}^\infty E^{\otimes m}$ which is the Lie algebra of the free nilpotent group G^n of n steps and $\mathcal{L}((E))$ the projective limit of the $\mathcal{L}^{(n)}(E)$ which is the Lie Series. Thus, we could easily check out the signature belongs to $T((E))$ and log signature belongs to $\mathcal{L}((E))$[Lyo14].
From last part, we know the signature could be expressed as

$$S(X)_{a,b} = 1 + \sum_{k=1}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} S(X)_{a,b}^{i_1,\dots,i_k} e_{i_1} \otimes \cdots \otimes e_{i_k}. \tag{2.44}$$

Thus, the log signature could be computed as:

Remark 4. *In order to use the Lie-bracket above, we need to find the coefficient of each Lie-bracket,*

$$logS(X)_{a,b} = \sum_{n \geqslant 1} \frac{(-1)^n}{n} (1 - S(X)_{a,b})^{\otimes n}. \tag{2.45}$$

For e_i part, it is easy to find the coefficient as $S(X)_{a,b}^i$ for $i = 1, ..., d$, and for the second $[e_i, e_j]$ part, we need to find out the coefficient about $e_i e_j$ and $e_j e_i$ as:

$$-(-S(X)_{a,b}^{i,j} e_i e_j - S(X)_{a,b}^{j,i} e_j e_i)^1 + \frac{1}{2}(-S(X)_{a,b}^i e_i - S(X)_{a,b}^j e_j)^2 = A, \tag{2.46}$$

where A is from (2.13) To get the above line, do not forget

$$(S(X)_{a,b}^i e_i) \otimes (S(X)_{a,b}^j e_j) = S(X)_{a,b}^{i \sqcup\!\sqcup j} e_i e_j. \tag{2.47}$$

Then, we could compute the log signature as:

$$logS(X)_{a,b} = \sum_{i=1}^{d} S(X)_{a,b}^i e_i + \sum_{1 \leqslant i < j \leqslant d} A[e_i, e_j] + \cdots. \tag{2.48}$$

The coefficients of the first few terms in above example leads to the Campbell-Baker-Hausdorff formula [CK16]:

Theorem 2. *We consider a path $X : [a, b] \mapsto \mathbb{R}$. Then we could express its log signature by real numbers $\lambda_{i_1,\dots,i_k}$:*

$$logS(X)_{a,b} = \sum_{k=1}^{\infty} \sum_{i_1,\dots,i_k \in \{1,\dots,d\}} \lambda_{i_1,\dots,i_k} [e_{i_1}, [e_{i_2}, ..., [e_{i_{k-1}}, e_{i_k}]...]]. \tag{2.49}$$

Note that $\lambda_{i_1,\dots,i_k}$ are not unique because the polynomials of the form are not linearly independent ($[e_1, e_2] \neq -[e_2, e_1]$).

2.4 Relation between Signature and rough path with path uniqueness

2.4.1 Signature and rough path

Above we state the basic information of signature method and the properties of it, which, to be more scientific and theoretical say, lays on the condition of paths which are piecewise differentiable or generally bounded variation. This is based on we use the normal Riemann-Stieltjes integrals to calculate the integration, but for more general cases, we use the Young integral for any path of finite p-variation with $1 \leqslant p < 2$ which could cover a class of paths precisely more irregular than those of bounded variation[CK16].
In essay [Lyons98], T. Lyons found that paths of infinite p-variation for all $p < 2$ has no well-defined notion of an iterated integral. And he gave a solution as if we could define the first $\lfloor p \rfloor$ iterated integrals of that kind of paths, then a unique way is existed to capture all the other iterated integrals. This finite p-variation path with its first $\lfloor p \rfloor$ iterated integrals is called a p-rough path[CK16].
Along the theory of rough paths, we can figure out there is an essential statement known as the universal limit theorem, which is able to give meaning to a controlled-DE where the driver belongs to a special class of, as above, p-rough paths, and if the solution depends continuously on the driver, the space which contains p-rough paths is equipped with a suitable topology, to be more specific, the p-variation metric, which means that there is still a large amount of research to be done in this area[CK16].

2.4.2 Uniqueness of Signature and rough path

As we have said, the signature obtains the shape of a path, so it is obvious to ask is the path totally determined by its signature or not. Looking back to the properties of signature, we say it is not determined completely by signature because of Chen's identity, the time-reversal property, as well as the invariance time reparametrizations[CK16]. An easy example, we cannot know the exact speed path is traversed from the signature because of the time reparametrizations, or no one can separate the trivial constant path and a path concatenated with its time-reversal because the signature of both is 1 as Chen's identity.
Another example is for a path crosses itself, the signature could not precisely describe the shape and the direction of the path, i.e. we would not figure out what happens between point to point if there is a cross. Thus these are still the challenge in the development of the signature method with rough path[CK16].
As for the uniqueness, essay [GGL+14] says that by Ben Hambly and Terry Lyons, signature determines the function of the gap process of the two-time point on a certain path and it is up to tree equivalent. So an important precondition of uniqueness is there is a sequence of $i \in 1, ..., d$ such that the path X_t^i is strictly (monotone) increasing. Such a bad condition and we need to do more work to check if there is a way to simplify it.

3 Practical machine learning application of signature method including codes

As we have discussed, the signature summarizes important feature about a path, then we could easily demonstrate numerical computations of signatures of certain data streams to use signature in machine learning problems, regression or classification[CK16].
During the next few parts, we discuss the elementary operations with signature transformation, the classification of time series and conclude the application of the signature method in machine learning.
About the current result of the signature method, kernelization of the signature has been tested and solved hand movement problem in figuring out Chinese character from the UCI

repository for machine learning[CK16].
We will use the iisignature package which is a Python package to calculate the iterated-integral signatures and log signatures of paths[RG18].

3.1 Elementary operations with signature transformation: Lead-lag transform

An important and interesting embedding is the lead-lag conversion of the data, which means that if we have a one-dimensional path, we can map it to a higher two-dimensional path. The reason we discuss the Lead-Lag transformation is that if we want to calculate the (log)signature of a path, we will implement it in Python using the iisignature package, which requires a two-dimensional path in the calculation function. Therefore, we need to convert a discrete array into a two-dimensional continuous path.
As a way of embedding discrete data streams, the Lead-Lag transform accounts for the relationship between the follow-up elements of the streams, which relatively perfect to reveal the information hidden in the path which has crossed data [KSH+16].
In order to express the Lead-Lag transform, we use a simple example[CK16]

Example 4. *Considering the sequence:*

$$X_{i\ i=1}^{1\ 4} = 1, 4, 2, 6 \tag{3.1}$$

the Lead-Lag mapping is given by:

$$LeadLag : X^1 = 1, 4, 2, 6 \mapsto \begin{cases} X^{1,Lead} = 1, 4, 4, 2, 2, 6, 6; \\ X^{1,Lag} = 1, 1, 4, 4, 2, 2, 6. \end{cases} \tag{3.2}$$

and the resulting embedded path is presented as:

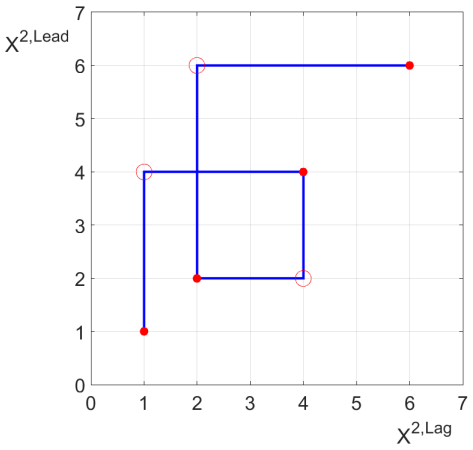

Figure 2: Lead-Lag path [CK16]

Simply checking from the equation above, there is an important property for lead-lag transform that we connect it with the signed area introduced in the above section and figure out that between the lead-stream data and the lag-stream data, the Lead-Lag transform obtain the quadratic variation of the data stream after some operation [KSH+16]. Thus, using this transformation, we could easily obtain the (log)signature of a certain path in Python, and then use this method to compute several applications.

3.2 The signature in machine learning

The main aim of computing signature for machine learning problems is we work for any type of sequential data which can be embedded into a continuous path and then use (log)signature to capture its feature. After that, we do the application like in (un)supervised learning or, to be more specific, one can classify time-series or distinguish clusters of data by signature method[CK16].
Considering the work-flow of how we use signature could be summarised as the following algorithm [CK16]:

$$data \to path \to signature\, of\, path \to features\, of\, data \tag{3.3}$$

There are quite a few advantages for using the Signature method, like it is sensitive to the geometric shape of a path, which has a great impact on application to Chinese character recognition problem. Another significant advantage is any time-ordered sequential data could compute its signature, like in quantitative finance field, we use data from institution to create a path, followed by computing the (log)signature and doing machine learning algorithms for further analysis. Thus, this method forms a new idea with data: regard data as geometric paths and using the path signature method in data analysis. We simply use a data model application to show how we use the signature method in data analysis during the next part. Additionally, there is a case that we input data from several independent parallel sources, after using the way of embedding data like the Lead-Lag or Time-Joined transformation, the path will result in a multi-dimensional case which involves observing the same quantity over periods again and again. An example for such is called "panel data" in financial or "longitudinal data" in medicine, biostatistics or other related areas [CK16]. In order to deal with these conditions, we need to find out a more efficient way of data processing.

3.2.1 Application of signature method in ARMA model analysis

In this section, we discuss a simple explicit example classification of time-series, which could be extended to sequential data of any type. Next section we will overview some of the recent applications in machine learning.
The mathematical model we use is a function which has the property of the rational transferring, which has various forms and is determined by estimates of the free parameters of the model. If all parameters are in the denominator of that function, the model is formed as an all-pole or autoregressive (AR) model, and for another condition, in the numerator, it represents all zeros or moving averages (MA) model. Thus, to state the definition of ($ARMA$) model is not a complex thing and it is defined as a model with free parameters in the numerator and denominator, called the pole-zero or ($ARMA$) model [PRT96].
As for its history, during 1951, Peter Whittle, formed a quite general model when he did the mathematical analysis for time-series; Box and Jenkins, made $ARMA$ model prevalent in 1970 by publishing a book contained the way to estimate the parameter of model[ARMA19]; the (Gaussian) $ARMA(p, q)$ model has such a covariance structure which is well understood by Priestley in 1981.
For a long time developing, it achieves a large number of applications in the partial analysis of real problems. As it is more efficient to predict time series based data sequence because of its ability to find a suitable forecasting model from the precondition of data, the $ARMA$ model is always used for understanding, and predicting trend in this series, especially for low-order ($n \leqslant 3$) polynomials. From above we know, ($ARMA$) models provide a brief description of a stationary stochastic process in terms of two parts, one for AR model which contains regression of variable from its own lagged values (time-delay term), with order p, another is MR model which contains the error term as a linear combination of past error terms occurring simultaneously or at different times, with order q. It is quite

accurate if the data stream is a combination of unobserved shocks and its own behaviour. For example, stock prices, because it may be influenced by basic market information as well as technology trends and mean regression effects caused by market participants, is a good data stream for $ARMA$ model.
We aim at forming a classifier which could discriminate between two types of same model time-series. The difference between $ARMA(p,q)$ time-series is in the model parameters. Consider sequences of time-series $Y_{i,j}$ labelled by two indices, where i for each distinct time-series and j for the measurement at time t_j[CK16].
We use $ARMA(1,1)$ model in this example and this Auto Regressive Moving Average model with parameters 1 (the order of the autoregressive model) and 1 (the order of the moving-average model) as below:

Definition 13. *The $ARMA(1,1)$ model has the form:*

$$(the autoregressive model) AR(1) : Y_t = \phi_0 + \phi_1 Y_{t-1} + \epsilon_t; \tag{3.4}$$

$$(the moving average model) MA(1) : Y_t = \mu + \epsilon_t + \theta_1 \epsilon_{t-1}. \tag{3.5}$$

where μ is the mean of the series and θ_i with ϕ_i are the parameters of the models and ϵ_{t-i} is the white noise error term.

Then we generate this model in Python with 1000 distinct time-series (500 of different parameters) of length 500. Following the codes from [JAI17], Giving a simple intuitive picture, the resulting different parameters path are:

```
import pandas as pd
import numpy as np
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
import statsmodels.stats as sms
import matplotlib.pyplot as plt

def tsplot(y, lags=None, figsize=(10, 8), style='bmh'):
    if not isinstance(y, pd.Series):
        y = pd.Series(y)
    with plt.style.context(style):
        fig = plt.figure(figsize=figsize)
        layout = (3, 2)
        ts_ax = plt.subplot2grid(layout, (0, 0), colspan=2)
        acf_ax = plt.subplot2grid(layout, (1, 0))
        pacf_ax = plt.subplot2grid(layout, (1, 1))
        qq_ax = plt.subplot2grid(layout, (2, 0))
        pp_ax = plt.subplot2grid(layout, (2, 1))

        y.plot(ax=ts_ax)
        ts_ax.set_title('Time Series Analysis Plots')
        smt.graphics.plot_acf(y, lags=lags, ax=acf_ax, alpha=0.05)
        smt.graphics.plot_pacf(y, lags=lags, ax=pacf_ax, alpha=0.05)
        sm.qqplot(y, line='s', ax=qq_ax)
        qq_ax.set_title('QQ Plot')
        scs.probplot(y, sparams=(y.mean(), y.std()), plot=pp_ax)

        plt.tight_layout()
    return

max_lag = 30
n = int(5000) # lots of samples to help estimates
```

```
burn = int(n/10) # number of samples to discard before fit
alphas = np.array([0.5, -0.25])
betas = np.array([0.5, -0.3])
ar = np.r_[1, -alphas]
ma = np.r_[1, betas]
arma22 = smt.arma_generate_sample(ar=ar, ma=ma, nsample=n, burnin=burn)
_ = tsplot(arma22, lags=max_lag)
```

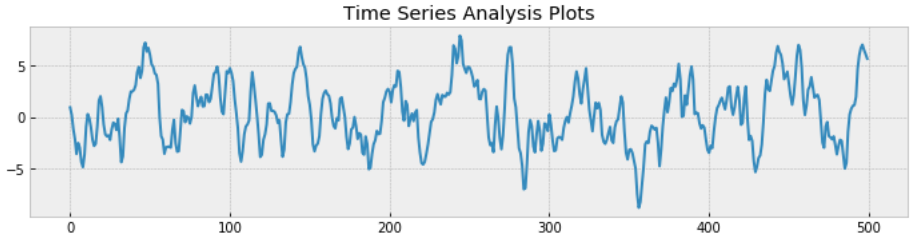

Figure 3: ARMA(1,1) path

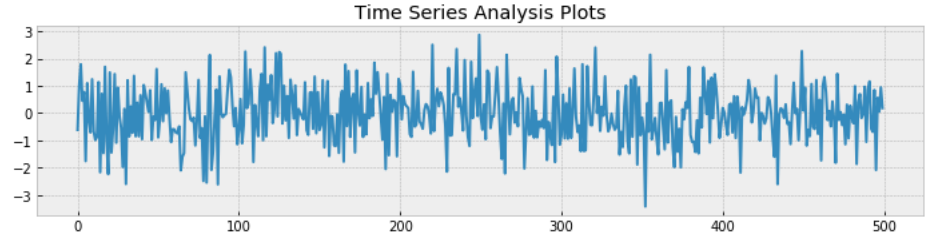

Figure 4: ARMA(1,1) another path

Then, We describe the essential steps [CK16]
• create the path X_i from each time-series Y_i row-wise;
• make use of the lead-lag transform to account for the variability in data;
• compute the (log)signature of the path X_i up to level L and use it to classify the different time-series.
Following is the self-made codes:

```
import pandas as pd
import numpy as np
import statsmodels.tsa.api as smt
import statsmodels.api as sm
import scipy.stats as scs
import statsmodels.stats as sms
import iisignature
import matplotlib.pyplot as plt

n = int(500)
burn = int(n/10)
alphas = np.array([0.5])
betas = np.array([0.4])
ar = np.r_[1, -alphas]
ma = np.r_[1, betas]
log = []
log = np.array(log)
sig = []
sig = np.array(sig)
```

```
for q in range(500):
    arma11 = smt.arma_generate_sample(ar=ar, ma=ma, nsample=n, burnin=burn)
    path = []
    path = np.array(path)
    for i in range(len(arma11)-1):
        kkk = ([arma11[i],arma11[i]],[arma11[i],arma11[i+1]])
        path = np.append(path,kkk)
    final = ([arma11[n-1],arma11[n-1]])
    path = np.append(path,final)
    finalpath = np.resize(path, (len(path),2))

    signature = iisignature.sig(finalpath,2)
    s = iisignature.prepare(2,2)
    logsignature = iisignature.logsig(finalpath ,s)
    sig = np.append(sig,signature)
    log = np.append(log,logsignature)
sig = np.resize(sig, (len(sig),6))
log = np.resize(log, (len(log),3))

#another 500 samples
alphas1 = np.array([0.8])
betas1 = np.array([0.1])
ar1 = np.r_[1, -alphas1]
ma1 = np.r_[1, betas1]
log1 = []
log1 = np.array(log1)
sig1 = []
sig1 = np.array(sig1)
for q in range(500):
    arma11ver2 = smt.arma_generate_sample(ar=ar1, ma=ma1, nsample=n, burnin=burn)
    path1 = []
    path1 = np.array(path1)
    for i in range(len(arma11ver2)-1):
        kkk1 = ([arma11ver2[i],arma11ver2[i]],[arma11ver2[i],arma11ver2[i+1]])
        path1 = np.append(path1,kkk1)
    final1 = ([arma11ver2[n-1],arma11ver2[n-1]])
    path1 = np.append(path1,final1)
    finalpath1 = np.resize(path1, (len(path1),2))

    signature1 = iisignature.sig(finalpath1,2)
    s = iisignature.prepare(2,2)
    logsignature1 = iisignature.logsig(finalpath1 ,s)
    sig1 = np.append(sig1,signature1)
    log1 = np.append(log1,logsignature1)
sig1 = np.resize(sig1, (len(sig1),6))
log1 = np.resize(log1, (len(log1),3))

#draw the plot
fig= plt.figure()
pl= fig.add_subplot(111)
pl.scatter(sig[:,3],sig[:,4],c = 'y')
pl.scatter(sig1[:,3],sig1[:,4])
pl.scatter(log[:,0],log[:,2],c = 'y')
pl.scatter(log1[:,0],log1[:,2])
```

In codes, we calculate the signature and the log-signature of each path and we get the graphs of $S(X_i)^{1,2}$ and $S(X_i)^{2,1}$ and $logS(X_i)^1$ and $logS(X_i)^{[1,2]}$ as below:
In order to find the difference caused by positive and negative signs of these four parameters, let us divide the group by the numbers of negative sign as:

$$\{0.5, 0.4, 0.8, 0.1\}, \{0.5, 0.4, -0.8, 0.1\}, \{0.5, 0.4, -0.8, -0.1\},$$

$$\{0.5, -0.4, 0.8, -0.1\}, \{-0.5, -0.4, -0.8, 0.1\}, \{-0.5, -0.4, -0.8, -0.1\}.$$

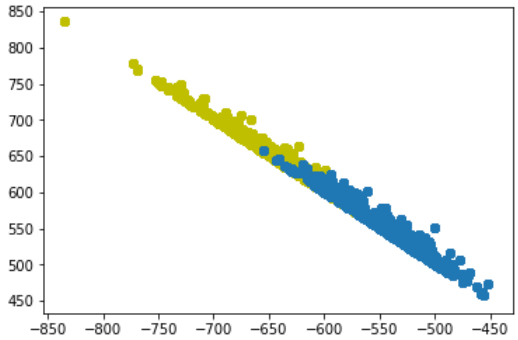

Figure 5: signature with para 0.50.40.80.1

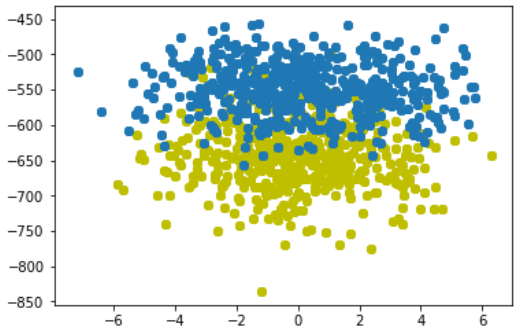

Figure 6: log-signature with para 0.50.40.80.1

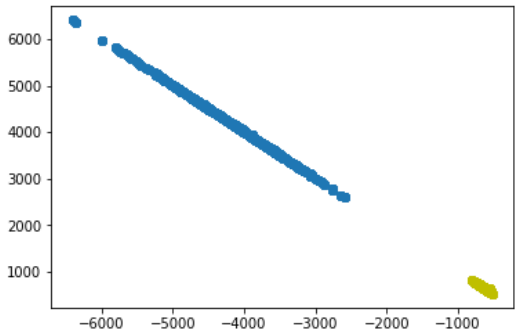

Figure 7: signature with para 0.50.4-0.80.1

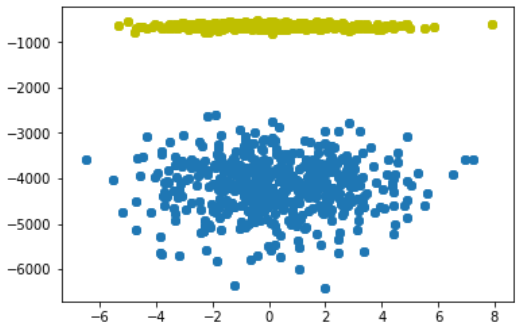

Figure 8: log-signature with para 0.50.4-0.80.1

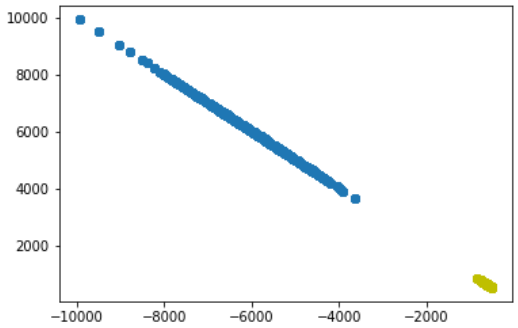

Figure 9: signature with para 0.50.4-0.8-0.1

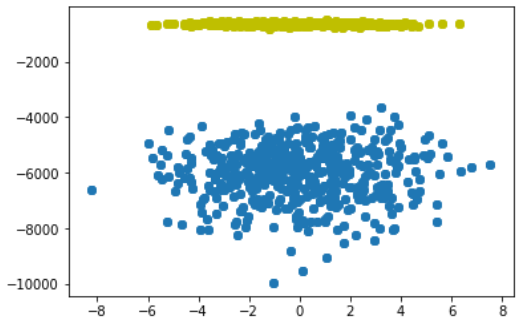

Figure 10: log-signature with para 0.50.4-0.8-0.1

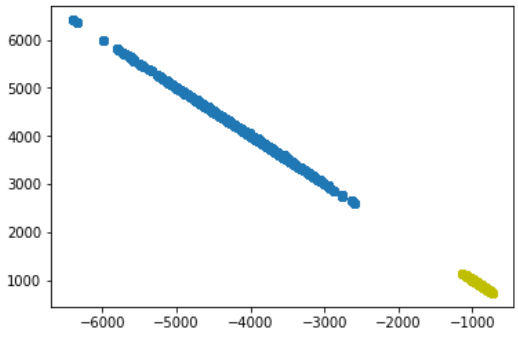

Figure 11: signature with para 0.5-0.40.8-0.1

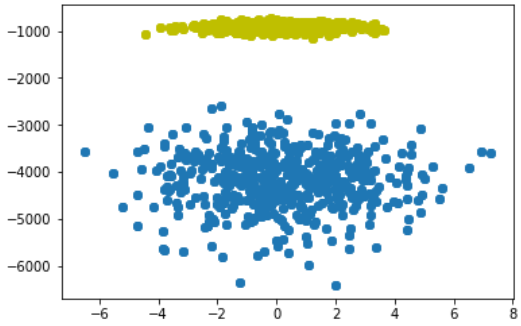

Figure 12: log-signature with para 0.5-0.40.8-0.1

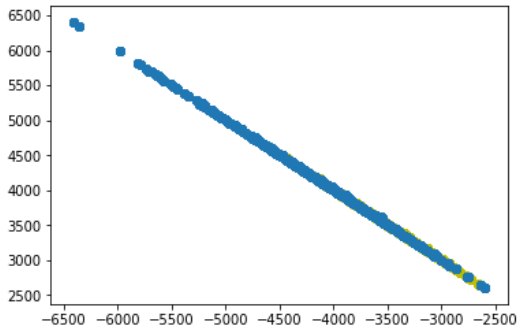

Figure 13: signature with para -0.5-0.4-0.80.1

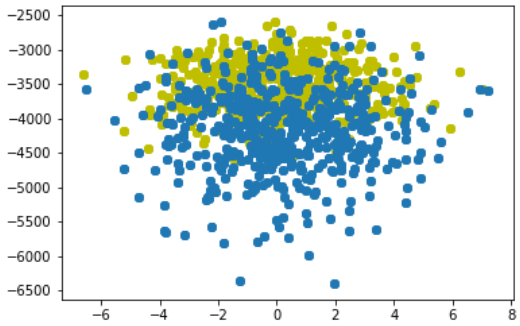

Figure 14: log-signature with para -0.5-0.4-0.80.1

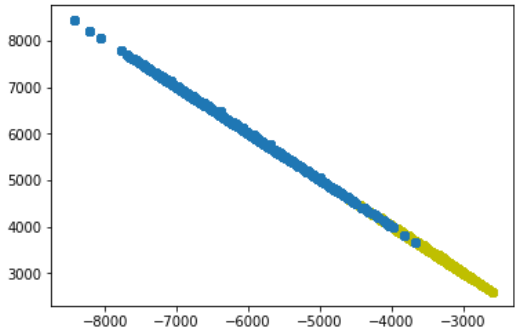

Figure 15: signature with para -0.5-0.4-0.8-0.1

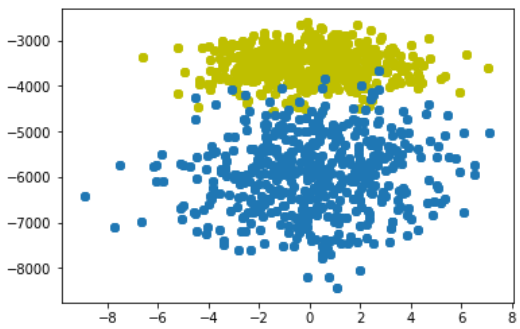

Figure 16: log-signature with para -0.5-0.4-0.8-0.1

From graphs, we get such ideas:
• For the all positive and all negative signs paras, they have the same law except the different position;
• for there is(are) one and two negative signs paras, they have quite the same graphs and that means both signature and log-signature could separate time-series quite clearly;
• something happens differently in three negative signs paras graph, it seems we cannot use any of these two features to separate which need more review and research (it may about the model and the stochastic error).
Since that, we could easily figure out that from graphs, different parameters have different graphs but similar conclusion: the signature or the log-signature could separate different time-series and we could use this as a sample to train the other data for deeper use in machine learning. The only thing we need is calculating the signature and the log-signature of each path. So in conclusion, this is quite an efficient method in the machine learning field.

3.2.2 Overview some of the recent applications of signature method in machine learning

The application areas are quite wide, including financial data, sound compression, time-series analysis, medical data, and image recognition tasks[CK16].
Like in financial area, scientists have applied the signature method to data streams coming from market to find hidden patterns in order to improve trading strategies, and it captures the properties of data in a non-parametric way without traditional statistical modeling; like the application of signature method with rough paths theory when solving the problem of sound compression and the applications of signature method combined with deep neural networks in the field of image recognition (the Chinese character); also like other medical and data analysis applications[CK16], all these make us feel that the signature method could achieve more with the machine learning.
In order to state much more clearly about the partial applications, we use • to divide up the different areas and arrange the statement by the information from [CK16] and [Lyo14]:
• it is well-done in the classification of time-buckets from standardised data:
This is for classification of the time of day by looking at the normalised data which is formed as a "buckets", a set of recorded normalised one-minute financial market data in 30 minutes intervals. For the methodology, we could use the low degree coordinates of market data's signature as a feature, and use least squares on the sample set to approximately rebuild function as a "classifier" and then test it on the out-of-sample set, which is quite efficient.
• it is useful in dealing with missing data in time-series:
There are many ways to deal with missing data which is definitely happened every day, like ameliorating imputing, taking sophisticated interpolation methods or Gaussian smoothing kernels and so on. Probably they would have a wonderful performance but actually, imputing missing values may directly change the data's underlying information and thus we need a lot more research in creating biased analysis which is still lots of work. As what we introduce now is the signature framework could naturally allow us to add a row in the indicator matrix, called index vector, into the path by changing the path into a higher dimension. For example, consider a data set contains missing values and its corresponding index vector, and we form a path of data. Then we create the "evolution" of the path as starting towards the end which is used as a propagation (time) to extend the third dimension, and the other two is observed data and missing data. At where there is a missing point, we jump from the first dimension to the second and taking the same value as the point just before the missing point unless we have any new information about the data. This approach allows us to process the data stream using the same unobserved data

as the full data stream, which will show another advantage of signature[CK16].
• it is so hot in taking out information from a financial data stream:
Several scientists have applied the signature to find hidden patterns in financial trading strategies because of its ability to recreate data just finding the signature of just a small set of original data except using traditional statistical modeling which is much more complex. In the article, the author aimed to distinguish orders generated by two different trade execution algorithms by using the computer to learn a discriminant function classifies streams based on their features from the signature.
They use linear regression with $LASSO$ to select terms related in the set of signature. They used non-parametric K-S test to measure the significance of this classification. As they have concluded signature method is one of the most useful ways to deal with financial data problems.
• it is also prevalent in the field of sound compression as connecting with the rough path:
T. Lyons and N. Sidorova worked on rough paths theory connected with the problem of sound compression in 2005 and they found there is a new way combined with the signature method is much more efficient than old Fourier and wavelet transforms. The difference lies on the linearity of the Fourier approach, such as signature collects non-linear dependencies but the old one not.
They said considering a situation of if we need to make a digital compression of a continuous with multi-dimensional area signal data which could be formed as a continuous piece-wise linear path, we could use the algorithm to compute the signature which could be proved as a representative of the coefficients of compression [CK16]. That is shown to be successful in the industry and reduce lots of work inside.
• it recently achieve great success in character recognition:
As we know, the signature could capture paths' feature which is quite suitable for image recognition. Mentioned in the first paragraph, the Chinese character is one of the ideal candidates because definitely it can be seen as continuous paths, and signature could be an efficient way to classify that. B. Graham and J. Lianwen did contributions in this area and they found that in a convolution neural network, the signature becomes more precisely to on-line character recognition, which achieved lots of awards around the world. Based on the signature method to research on the image indicate signature method combined with deep neural networks is regarded as an effective way of image recognition[CK16].
• it may have a wonderful use in studying from the past system and predicting the statistics features for the future:
As essay [CK16] said, Levin, Lyons, and Ni found a new regression method in the field of time-series based on the expected signature of paths, which is much more general and categorically separating computations and predictions. Their aim is to identify the specific feature (excepted signature) of the experimented data and realize the functional relationships between features. It essentially means that there are numerical benchmarks between different AR models and set up the various expected signature model for real problems.
• it is applied in identifying patterns from MEG scans:
Gyurko has devoted to the pattern recognition by using signature in medical studies and neuron-imaging(MEG). He aimed to identify a pattern in data streams and in order to simplify it, he considered only two channels from the MEG scanner as the original signal which could be used signature method to predict the pressing activity of a button (shown as a vector of $\{0, 1\}$ as outcome). What they have done is using a rolling window to analyse time-series and while inside the rolling window, calculate the signature of regressed button-pressing cases[CK16]. We can see there is no limitation of the field we use signature unless we are not focusing on classifying or predicting.
• it is functional to understand the impact of treatment on behavioural patterns in bipolar disorder patients:
Essay [CK16] showed an experiment they have done as learning the behavioural patterns of patients suffering from bipolar disorder. What they analysed is the delays: the time

intervals between the prompt text which is formed after patients submitting their self-test scores to the "True Colours platform" weekly and the patient's reply measured in integer units of days. For missing observation, the system would send a new prompt next week beginning. The author said that they focused on whether the distribution or law of the delays is different between two separate treatment groups. With considering the signature of the time-series formed from the delay data and using the regularised logistic regression, they found the figure of signature could directly separate the difference and this demonstrated the ability of the signature to capturer features in a systematic way and after that, the features could be served as an initial inputs in the machine learning.

4 Rooted tree

In this section, we turn back our focus at the theoretical part of the signature method, as we have through is this method with rough path and this part, we pay more attention on it with the branched rough path like forest or tree. Firstly we discuss the basic information and history of the rooted tree and then we try to use the B-series to write the "Picard iteration" of the forest.

4.1 History of rooted tree

We will firstly introduce the tree, which was coined from Arthur Cayley in 1857, in graph theory which is the basic founding of rooted tree. A tree is a direction-less graph of any two points inside are connected by exactly one path, or, to be more simple, a connected off-cycle undirected graph in graph theory. For a forest, is similarly defined as a direction-less graph of any two points inside are connected by at most one path, or, to be more simple, a disjoint union of trees [Tree19].
There are various kinds of trees as data structures in computer science which is formed by basic trees in graph theory, but in fact, they are always "rooted trees". A difference between "rooted tree" and "general tree" is "rooted tree" may have direction, and there are conditions either its edges point all away from the root or they are all towards the root, naming differently. Also, there is a definition of the rooted forest as a disjoint union of rooted trees. Quite similar as the rooted tree, the forest may be directed and either in each rooted tree contained, all edges point away from the root, or, towards the root, namely different as well [Tree19].
Due to Butcher's work, rooting trees are considered to be the basic mathematical structure used in the numerical approximations or exact solutions of ODE. This mathematical structure is also shown in the task of Connes and Kreimer, about renormalizing it and its Hopf algebra structure. Kreimer also introduced nested integrals indexed by trees and Hoffman, Foissy did several workings on labeled rooted trees[GUB10].
Let consider this question from the idea of "change rough to branched rough path" in order to discuss it more logically. Deeply considering the essence of the algebra structure of rough path is isomorphic with the Hopf algebra using Chen's iterated integrals theory [HK15]. Feyel de La Pradelle has done some researches on alternative method for rough paths and this is what we mention about "is the basic foundation of T. Lyons's rough path theory to recover series solutions of DE driven by such a path".
Why we consider the rooted tree as what we call it "branched rough tree", it may because for we generally thought is a differential equation driven by a rough path, it may be not an appropriate model for a stochastic system, especially in some financial model which use the $Itô$ integral since we need to look into future, and also the formal approximations of stochastic integrals could not converge to objects for which hold the general change of variables formula. To be more specific, we figure out the discrete approximations to an integral not satisfying the "integration by parts" formula. Thus, if we want to integrate a

path against another, the resulting error term would vanish but the stochastic case cannot. So, it is easy to obtain that discretization schemes outputs are called as "non-geometric", which is stated as "branched rough path" by M. Gubinelli [HK15]. Clearly, for developing during years, the index iterative integration of the tree creates a closed algebra consisting of a point-by-point product and a path integral. Thus, if we could just enrich the notion of the rough path, it is available for us to link the computations of general rough paths, in order to form a complete theory to get solution of SDEs [GUB10].

4.2 Rooted tree

Definition 14. *The $\mathcal{L}$-labeled rooted tree is defined as a finite map with a starting vertex called the root such that there is a unique path from the root to any other vertex of the tree, and for each vertex on the rooted tree, there is a finite set of tokens $\mathcal{L}$ marked by side.*

For example, if the label set is $\mathcal{L} = 1, 2, 3$, there are several trees like:

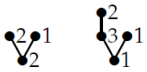

Figure 17: rooted tree example[GUB10]

Definition 15. *Define the operation $[\cdots]_a$ as a tree obtained by appending the root of the tree in parentheses to the new vertex with the label a, which will be the root of this new tree [GUB10].*

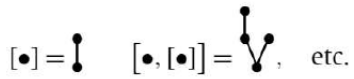

Figure 18: Operation of rooted tree [GUB10]

Thus, any decorated rooted tree can be formed using a simple decorative tree $\bullet_a$ which a is from the label set.
Denoted $\mathcal{T}_{\mathcal{L}}$ the set of all $\mathcal{L}$-labeled rooted tree and $\mathcal{T}$ the set of undecorated rooted trees.

Definition 16. *Let $|*| : \mathcal{T} \to \mathbb{R}$ the map calculates the number of vertices of rooted tree, and the tree factorial $\gamma : \mathcal{T} \to \mathbb{R}$ as:*

$$\gamma(\bullet) = 1, \gamma([\tau_1, ..., \tau_k]) = |[\tau_1, ..., \tau_k]|\gamma(\tau_1) \cdots \gamma(\tau_k). \tag{4.1}$$

Moreover, the symmetry factor is defined as $\sigma : \mathcal{T}_{\mathcal{L}} \to \mathbb{R}$ with the recursive formula $\sigma(\tau) = 1$ for $|\tau| = 1$ and

$$\sigma([\tau^1, ..., \tau^k]_a) = \frac{k!}{\delta(\tau^1, ..., \tau^k)}\sigma(\tau^1) \cdots \sigma(\tau^k). \tag{4.2}$$

Where $\delta(\tau^1, ..., \tau^k)$ represents the number of different ordered "$k-uples$"$(\tau^1, ..., \tau^k)$ corresponds to the same (unordered) set of sub-trees $(\tau^1, ..., \tau^k)$. The first part is the order of the subgroups used to calculate the k element, which does not change the order of $k-uple$, $\sigma(\tau)$ is the order of the subgroups arranged. The vertex of the tree τ does not change the tree (including the label) [GUB10].

Then we make a notation: we use $\mathcal{AT}_{\mathcal{L}}$ as the set of the commutative polynomial algebra with generator $\{1 \cup \mathcal{T}_{\mathcal{L}}\}$ over $\mathbb{R}$ and $\mathcal{F}_{\mathcal{L}}$ is for the collection of all tree monomials, called forests, including the empty forest 1 [GUB10].
Then, on the algebra $\mathcal{AT}_{\mathcal{L}}$, we have a definition of "*counit*" and "*coproduct*":

Definition 17. *A counit $\epsilon : \mathcal{AT}_{\mathcal{L}} \to \mathbb{R}$ is an algebra homomorphism with equation $\epsilon(1) = 1$ and $\epsilon(\tau) = 0$; A co-product $\Delta : \mathcal{AT}_{\mathcal{L}} \to \mathcal{AT}_{\mathcal{L}}$ is an algebra homomorphism and if expressed by "admissible cuts c"(select a subset of the edges of the tree to delete, and if starting from root to any leaf, we can satisfy up to one cut), the co-product is:*

$$\Delta(\tau) = 1 \otimes \tau + \tau \otimes 1 + \sum_{c} P_c(\tau) \otimes R_c(\tau) \tag{4.3}$$

where the sum is Eimstein convertion of summation over cuts, $R_c(\tau) \in \mathcal{T}_{\mathcal{L}}$ the tree obtained after the cut and $P_c(\tau) \in \mathcal{F}_{\mathcal{L}}$ the set of subtrees separated from the "trunk" because of the cut [GUB10].
There is a deformation of this co-product called reduced co-product which could be shown easily by examples:

$$\Delta(\tau)^{'} = \Delta(\tau) - 1 \otimes \tau - \tau \otimes 1 = \sum_{c} P_c(\tau) \otimes R_c(\tau) \tag{4.4}$$

Example 5. *Then, we would like to show some examples, from which it is easy for ue to figure out that the partial meaning for the reduced co-produce is the sum of the tree obtained after the cut and the other subtrees parts separated from the "trunk" because of the cut.*

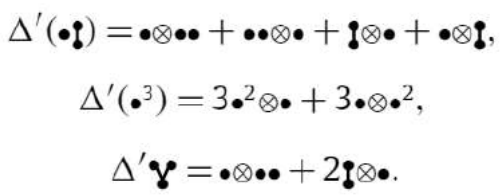

Figure 19: Reduced co-product example[GUB10]

Remark 5. *Combined with ϵ and Δ, the algebra $\mathcal{AT}_{\mathcal{L}}$ becomes a bialgebra.*

4.3 Basic information for iterated increment of rooted trees

In order to consider with the differential equation and the "Picard iteration", we need to know the iterated integral of branched rough path rooted tree.

Definition 18. *For a given $T > 0$, we denote the set of functions $g : [0, T]^k \to \mathbb{R}$ s.t. $g_{t_1,\dots,t_k} = 0$ when $t_i = t_{i+1}$ for some $0 \leqslant i \leqslant k-1$ as $C_k(\mathbb{R})$, written as C_k, and we call such g is a k-increment [GUB10].*

Definition 19. *There is a cochain complex $(\cup_{k \geqslant 1} C_k, \delta)$, and we define the common boundary δ as a derivative operator with equations:*

$$\begin{aligned} &\delta : C_k \to C_{k+1}, \delta^2 = 0; \\ &(\delta g)_{t_1,\dots,t_k} = \sum_{i=1}^{k+1} (-1)^i g_{t_1,\dots,\hat{t_i},\dots t_k} \end{aligned} \tag{4.5}$$

where $\hat{t_i}$ states that we ignore this argument.

Then, for a set of smooth elements $x = \{x^a\}_{a\in\mathcal{L}=\{1,...,d\}} \in C_1$, we iterate the integration along x, and build a map $X : \mathcal{T}_{\mathcal{L}} \to C([0,T]^2;\mathbb{R})$ as:

$$X_{ts}^{\bullet_a} = \int_s^t dx_u^a = \delta x^a, X_{ts}^{[\tau_1,...,\tau_k]_a} = \int_s^t \prod_{i=1}^k X_{us}^{\tau^i} dx_u^a. \tag{4.6}$$

For reason why we focus on C_2 is that we need to search about the iteration and it lays on the second vector space, and considering about the algebraic structure of C_2, we introduce the associative and commutative inner product $\circ$ which is similar as the "concatenation product" in Chen's identity part: $(a\circ b)_{ts} = a_{ts}b_{ts}$ for $a, b \in C_2$, and with this, C_2 becomes an algebra. The map X is extended to a map on $\mathcal{AT}_{\mathcal{L}}$ by linearly and also by the equation $X_{ts}^{\tau_1,...,\tau_n} = X_{ts}^{\tau_1} \cdots X_{ts}^{\tau_n}$ for the X values on the forest $\tau_1, ..., \tau_n$ [GUB10].

Remark 6. *With this product, we get:*

$$X^{[\tau_1,...,\tau_k]_a} = \int X^{\tau_1\cdots\tau_n} dx^a. \tag{4.7}$$

This is a product for elements in C_2, like a product for paths, and we consider a exterior product on the algebra $(C_2, \circ)$ as from $C_2 \otimes C_2$ to C_3, then we could extend X to the tensor product between forest like from $\mathcal{AT}_{\mathcal{L}} \otimes \mathcal{AT}_{\mathcal{L}}$ to C_2 as: $X^{\theta\otimes\rho} = X^\theta X^\rho$ for all $\theta, \rho \in \mathcal{AT}_{\mathcal{L}}$ [GUB10].
Above we discuss the basic of iterated integration and its algebra structure and next part is for consider the series solution of a differential equation driven by forest, or, more precisely , the branched rough path.

4.4 Picard iteration of rooted tree

From essay [GUB10], we know for the vectorfield $f : \mathbb{R}^n \to \mathbb{R}^n$, if it is under appropriate conditions, we could get the series representation solution of a differential equation:

$$\frac{dy}{dt} = f(y), y_0 = \xi, \tag{4.8}$$

as:

$$y_t = \xi + \sum_{\tau\in\mathcal{T}} \psi^f(\tau)(\xi)\frac{t^\tau}{\sigma(\tau)\tau!}, \tag{4.9}$$

which is called as a B-series representation in order to memory J. Butcher, and the coefficients ψ^f are called "elementary differentials" as:

$$\begin{aligned} \psi^f(\bullet)(\epsilon) &= f(\epsilon), \\ \psi^f([\tau^1,...,\tau^k])(\epsilon) &= \sum_{k=0}^{\infty}[\prod_{i=0}^k \sum_{b_i} \partial_{b_i} f(\epsilon)][\prod_{i=0}^k \sum_{b_i} (\psi^f(\tau^i)(\epsilon))^{b_i}], \end{aligned} \tag{4.10}$$

where we need to explain when $k, i = 0$, the b_i is ϕ, the ∂_ϕ is itself for which does not take the derivation, the function $(\psi^f(\tau^i)(\epsilon))^\phi$ is general 1 and the sum is on the b_i for which $i \in \mathcal{L}$
Then, let us consider a driven differential equation: we have a C_1 path $x : [0,T] \to \mathbb{R}$, and let the set $\{x^a\}_{a\in\mathcal{L}=\{1,...,d\}}$ be the fixed basis. Considering an initial point $\xi \in \mathbb{R}$ and a function $f_a : \mathbb{R} \to \mathbb{R}$, we have a theorem as followed:

Theorem 3. *For the result of the differential equation:*

$$dy_t = \sum_{a\in\mathcal{L}} f_a(y_t)dx_t^a, y_0 = \xi, \tag{4.11}$$

we admit the series representation as:

$$\delta y_{ts} = \sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} \phi^f(\tau)(y_s) X^{\tau}_{ts}, y_0 = \xi, \tag{4.12}$$

where we define $\phi^f : \mathcal{T}_{\mathcal{L}} \times \mathbb{R}^n \to \mathbb{R}^n$ *as:*

$$\begin{aligned} \phi^f(\bullet_a)(\epsilon) &= f_a(\epsilon), \\ \phi^f([\tau^1, ..., \tau^k]_a)(\epsilon) &= \sum_{k=0}^{\infty} [\prod_{i=0}^{k} \sum_{b_i} \partial_{b_i} f_a(\epsilon)][\prod_{i=0}^{k} \sum_{b_i} (\phi^f(\tau^i)(\epsilon))^{b_i}], \end{aligned} \tag{4.13}$$

where the $k, i = 0$ *case is what we have explained above and we sum up from all direction of* $\mathbb{R}^d$ *[GUB10].*

Proof. We need to verify that (4.12) satisfy the integral equation:

$$\delta y_{ts} = \sum_{a \in \mathcal{L}} \int_s^t f_a(y_u) dx^a_u, \tag{4.14}$$

Then we consider the Taylor series for f at point $\epsilon \in \mathbb{R}^n$:

$$f_a(\epsilon^{'}) = \sum_{k=0}^{\infty} [\prod_{i=0}^{k} \sum_{b_i} \frac{\partial_{b_i} f_a(\epsilon)}{k!}][\prod_{i=0}^{k} \sum_{b_i} (\epsilon^{'} - \epsilon)^{b_i}] \tag{4.15}$$

where the $k, i = 0$ is the case we explain above and ϵ^k is the k-th coordinate of the vector ϵ.
Next is we need to check the right side of (4.14) which could be rewritten by plugging the right side of (4.12):

$$\begin{aligned} \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \int_s^t f_a(y_u) dx^a_u &= \sum_{a \in \mathcal{L}} \prod_{i=0}^{k} \sum_{b_i} \int_s^t \frac{\partial_{b_i} f_a(y_s)}{k!} \delta y^{b_i}_{us} dx^a_u \\ &= \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \prod_{i=0}^{k} \sum_{b_i} \frac{\partial_{b_i} f_a(y_s)}{k!} \int_s^t [\sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} [\phi^f(\tau)(y_s)]^{b_i} X^{\tau}_{us}] dx^a_u \\ &= \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \prod_{i=0}^{k} \sum_{b_i} \frac{\partial_{b_i} f_a(y_s)}{k!} \int_s^t [\sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} [\phi^f(\tau)(y_s)]^{b_i} X^{\tau}_{us}] dx^a_u \\ &= \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \prod_{i=0}^{k} \sum_{b_i} \frac{\partial_{b_i} f_a(y_s)}{k!} \sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} ([\phi^f(\tau)(y_s)]^{b_i}) \int_s^t X^{\tau}_{us} dx^a_u \\ &= \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \prod_{i=0}^{k} \sum_{b_i} \frac{\partial_{b_i} f_a(y_s)}{k!} \sum_{\tau^i} \frac{1}{\sigma(\tau^1) \cdots \sigma(\tau^k)} ([\phi^f(\tau)(y_s)]^{b_i}) X^{[\tau^1, ..., \tau^k]_a}_{ts} \\ &= \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \prod_{i=0}^{k} \sum_{\tau^i} \frac{1}{k! \sigma(\tau^1) \cdots \sigma(\tau^k)} \sum_{b_i} \partial_{b_i} f_a(y_s) ([\phi^f(\tau)(y_s)]^{b_i}) X^{[\tau^1, ..., \tau^k]_a}_{ts} \\ &= \sum_{a \in \mathcal{L}} \sum_{k=0}^{\infty} \sum_{\tau^i} \frac{1}{\sigma([\tau^1, ..., \tau^k]_a) \delta(\tau^1, ..., \tau^k)} \phi^f([\tau^1, ..., \tau^k]_a)(y_s) X^{[\tau^1, ..., \tau^k]_a}_{ts} \\ &= \sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} \phi^f(\tau)(y_s) X^{\tau}_{ts} \end{aligned} \tag{4.16}$$

□

Thus, we figure out a series representation we called B-series as the series solution of

a differential equation driven by rooted tree.
Then similarly as what we have done in part two, we could state what we call the "Picard iteration" of computing the solution of differential equation driven by rooted tree as:
For the equation (4.11), the arbitrage path $y : [0, T] \to \mathbb{R}$, and all $t \in [0, T]$, we use the B-series to have followed iteration:
Let us define a new path $F(y)_t$ from the equation (4.11):

$$F(y)_t = \xi + \sum_{a \in \mathcal{L}} \int_s^t f_a(y_u) dx_u^a \tag{4.17}$$

Easily check out that y is a solution to (4.17) iff y is a fixed point of F, then we consider the sequence of paths $y_t^n = F(y^{n-1})_t$ with initial arbitrary path $y_t^0 = \xi$ and if assumption suitable, F leads to a unique fixed point y and that y_t^n converges to y as n goes to infinity. Then the iterates of F can be expressed as follows:

$$y_t^0 = y_0 = \xi; \tag{4.18}$$

$$\begin{aligned} y_t^1 &= \xi + \sum_{a \in \mathcal{L}} \int_s^t f_a(y_t^0) dx_u^a \\ &= \xi + \delta y_{ts} = \xi + \sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} \phi^f(\tau)(y_t^0) X_{ts}^\tau; \end{aligned} \tag{4.19}$$

$$\begin{aligned} y_t^2 &= \xi + \sum_{a \in \mathcal{L}} \int_s^t f_a(y_t^1) dx_u^a \\ &= \xi + \delta y_{ts} = \xi + \sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} \phi^f(\tau)(y_t^1) X_{ts}^\tau; \end{aligned} \tag{4.20}$$

$$\vdots$$

$$\begin{aligned} y_t^n &= \xi + \sum_{a \in \mathcal{L}} \int_s^t f_a(y_t^{n-1}) dx_u^a \\ &= \xi + \delta y_{ts} = \xi + \sum_{\tau \in \mathcal{T}_{\mathcal{L}}} \frac{1}{\sigma(\tau)} \phi^f(\tau)(y_t^{n-1}) X_{ts}^\tau. \end{aligned} \tag{4.21}$$

$$\vdots$$

Thus, we could take the limitation of the n-th equation and get the approximate series solution of a certain differential equation. In conclusion, we compute the series solution of differential equation by "Picard iteration" and the B-series using its driven rooted tree.

5 Conclusion

This part is for summaries of this essay and extends what should be done next.

5.1 For signature

Since it is formed from the last century, the theory of rough path and its signature becomes active in recent year, especially after combining with machine learning. We do believe this is a brilliant way for us to capturer the feature of a data stream because of its well-defined algebra structure. It starts with regarding as one of the most efficient ways for the first step of machine learning. The whole algorithm for computing the signature is quite strict-forwards, with the first step collecting the data, the second step forming a path either

using Lead-Lag or Time-Joint transform, and the third step calculating (log)signature by code package, which is the input of a machine learning project. Seems quite direct and easy-understanding, but actually it has great use in classification and recognition, like data analysis, medical confirmation, or financial predicting so on. It could not be ignored as a significant power in the data process.
What may be the further area for signature is how to improve the accuracy of signature, as we only use the first few levels of signature, it has a great impact if we could get a more accurate result. And also, for my application example in the second part, there is an unusual situation if I change the model parameters, so it should be the fault of noise because thousands of experiments show thousands of graphs finally. Thus, is it available for us to modify the code of calculating the integration to fix the result of our graph may be the further task for me.
All in all, the signature method is quite useful and widely influence the algorithm of machine learning. Do believe it will achieve more in the applied mathematics as well as other scientific fields.

5.2 For rooted tree

As an area combined the graph theory and differential equation analysis, the tree has formed a high reputation in solving the equation, which is not an easy-understanding area. In 2010, M. Gubinelli is the person who made the theory of rooted tree up to a higher level, forming several algebras to introduce its numerical meaning, also gave the chance to connect with the geometric rough path and its signature. This is benefited by the new integration method and an idea of forming a complete field of the rough path. As for its "(log)signature", there is same algebra structure for both branched rough path or rough path, but for branched one, it seems not quite useful in the algebra of log-signature because of the property is the same as itself. Also, for my work, changing the notation of what Gubinelli has said is not quite strict-forwards because he used lots of notes combined the algebra and the tree. Independently forming the "Picard iteration" makes me feel more understandable about both the original one and the higher-level one. Further work is needed to study the algebraic background of rooted trees, which for me will be understood in essence. All in all, for the rough path of the branch, I only touched the surface of the surface, so I would be happy to continue my research if I can.

Acknowledgement

First of all, I am very grateful to my academic director Yvain Bruned for his patient guidance and constructive advice. He gave me good guidance and encouragement throughout the entire process of reading papers, selecting topics and completing writing. His insights into each manuscript provided me with many inspiring ideas and introduced me to a profound world of mathematics. It is no exaggeration to say that this article cannot be completed without his patient guidance.
Secondly is my parents, families and friends, every people around me. My parents always comfort me to not be too worry and my girlfriend is quite empathetic when it is not available for me to do some daily cooking. It is not available for me to go through this period smoothly without them.
The third is my lecturers who helped and taught me during the year, which greatly broadened my horizons and enriched my knowledge. Their inspiring and responsible teaching provides a solid foundation for the writing of this article and is of great value to my future academic research.
Finally is all the authors of my references, feel honour to review their essay and step on

what they have done before, and also it is a great experience to look through the developing history, feeling lofty respect to these people.

References

[Lyo14] T. Lyons. Rough paths, Signatures and the modelling of functions on streams. *ArXiv e-prints* (2014). arXiv:1405.4537.

[YLNSJC17] Yang, Weixin, Lyons, Terry, Ni, Hao, Schmid, Cordelia, Jin, Lianwen, and Chang, Jiawei. Leveraging the Path Signature for Skeleton-based Human Action Recognition. *arXiv preprint* (2017). arXiv:1707.03993.

[Lyo07] Lyons, Terry J, Caruana, Michael, and Lévy, Thierry. In *Differential Equations Driven by Rough Paths*. Springer, 2007.

[Ni14] Hao Ni. A multi-dimensional stream and its signature representation. *ArXiv e-prints* (2015). arXiv:1509.03346.

[RG18] Jeremy Reizenstein, Benjamin Graham. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *ArXiv e-prints* (2018). arXiv:1802.08252.

[CK16] Ilya Chevyrev, Andrey Kormilitzin. A Primer on the Signature Method in Machine Learning. *arXiv e-prints*(2016). arXiv:1603.03788.

[ML19] Wikipedia. Machine learning. URL https://en.wikipedia.org/wiki/Machine_learning.

[JAI17] Shub Jain. Time Series Analysis for Financial Data IV— ARMA Models. URL https://medium.com/auquan/time-series-analysis-for-finance-arma-models-21695e14c999.

[GUB10] Massimiliano Gubinelli. Ramification of rough paths. *J.Differ.Equ.248.* (2010)693-721

[C19] Christopher C. Tisdell. On Picard's iteration method to solve differential equations and a pedagogical space for otherness. *International Journal of Mathematical Education in Science and Technology* **50:5**(2019), 788-799. DOI: 10.1080/0020739X.2018.1507051.

[EML53] Samuel Eilenberg, Saunders Mac Lane . On the Groups H(∏, n), I *Annals of Mathematics Second Series* **Vol. 58**, No. 1 (July, 1953), pp. 55-106. DOI: 10.2307/1969820.

[Chen58] Kuo-Tsai Chen. Integration of paths – a faithful representation of paths by noncommutative formal power series. *Trans. Am. Math. Soc.* **89**, 1958, 395–407.

[ARMA19] Wikipedia. ARMA model. URL https://en.wikipedia.org/wiki/Autoregressive\OT1\textendashmoving-average_model.

[Tree19] Wikipedia. Tree (graph theory). URL https://en.wikipedia.org/wiki/Tree_(graph_theory).

[Lyons98] Terry J. Lyons. Differential equations driven by rough signals. *Rev. Mat. Iberoamericana* **14**, no. 2, (1998), 215-310.

[GGL+14] Lajos Gergely Gyurko, Terry Lyons, Mark Kontkowski, and Jonathan Field. Extracting information from the signature of a financial data stream. *ArXiv e-prints* (2014). arXiv:1307.7244.

[KSH+16] A.B.Kormilitzin, K.E.A.Saunders, P.J.Harrison, J.R.Geddes, T.J.Lyons. Application of the Signature Method to Pattern Recognition in the CEQUEL Clinical Trial. *ArXiv e-prints* (2016). arXiv:1606.02074.

[PRT96] J. Pardey, S. Roberts and L. Tarassenko. A review of parametric modelling techniques for EEG analysis. *Med. Eng. Phys.* **18**, no.1, (1996), 2-11.

[HK15] Martin Hairer, David Kelly. Geometric versus non-geometric rough paths. *Annales de l'Institut Henri Poincaré - Probabilités et Statistiques***Vol.51**, no.1, (2015), 207–251. DOI: 10.1214/13-AIHP564