Signatures of BV Paths and its Application to Online Handwriting Recognition

MATH 820 Advanced Topics in Analysis: Variational and PDE Techniques in Data Analysis
Center for ~~Non~~linear Analysis Seminar Room, December 14th $\otimes$ 16th, 2016

This report is produced per the course requirement of MATH 820 in Fall 2016 offered by Professo Dejan Slepcev. Its main goal is to introduce to fellow students the signature (sometimes called tensorial exponential) of a continuous path of bounded variations, and its application to analysis of data streams (i.e. one-parameter representations in a state space), with writer-independent online handwriting digit recognition as a prototype of concepts. With an emphasis on the big picture, the report is not intended to be rigorous or comprehensive, and in particular, theorems and propositions are not proved. Interested readers can refer to the citations for more details. The author claims no expertise in either signatures of paths (the abstract algebraic leg of rough path theory) or data analysis (especially handwriting recognition), but is responsible for any mistake not due to the references.

Contents

1 Tensor Algebra, the Signature of a Path, and Linearity

Description and prediction of dynamical systems such as weather forecasting, speech recognition, and financial investment are fundamental, intriguing, and difficult problems in both daily life and research, which have been studied by the most brilliant minds of human beings since as late as the advents of calculus and classical mechanics. A classical and currently dominant mathematical tool for such problems is ordinary differential equations (ODE), in which the infinitesimal change of the system with respect to time is described as a time-dependent function of the current state of the system and external forcing. However, ODE theory often requires suitable regularities of the system and forcing to produce meaningful outputs, and hence is limited in dealing with highly oscillatory and complex systems, for example systems of merely bounded variations, or signals corrupted by noises. A more general framework, called rough path theory, has recently been advocated by Lyons to address the lack of (classical) regularities in modeling evolving systems. According to Lyons [9] and Geng [1], rough path theory has proven successful ranging from commercial applications such as Chinese handwriting recognition by Graham, to theoretical developments such as regularity structures and stochastic partial differential equations (SPDE) by the Fields medalist Hairer 3. In this section, we shall sample taste the abstract algebraic leg of rough path theory, namely the signature of a path, or sometimes called the (tensorial) exponential map. Later in the next section, we shall experiment with handwriting data and apply the theory of signature to a mini-task of online handwriting digit recognition, in contrast to Graham [3]. We conclude this report with general remarks of signatures as extracted features in analyzing data streams.

Let's start with the mathematical formulation of paths, evolving systems, and/or data streams, as they appear in the introductory paragraph above.

Definition 1.1 (Path)**.** Let E be a set and I be a linearly ordered set. A path in E is a function $X : I \to E$. If E and I have additional mathematical structures such as σ-algebra, topology, or metric, it is usual to restrict X to be a homomorphism in some corresponding category. In natural science and engineering, it is common to refer X as an evolving system with state space E; in social science and data analysis, X is often called an E-valued data stream. In all these cases, I is interpreted as time.

In this report, we consider $E = \mathbb{R}^d$ for some $d \in \mathbb{N}_+$, $I = [a, b] \subset \mathbb{R}$ a compact interval, and X being continuous and of bounded variations, i.e. $X \in C \cap BV(I; E)$[1], without losing much generality in applications yet keeping the introduction accessible.

Next, we give the definition of the signature $S(X)$ of the path $X : I \to E$ along with its basic properties , and demonstrate its significance and advantages in data science (thanks to abstract algebra and category theory).

1.1 Tensor Algebra, Definition and Motivation of Signatures

We need some basics of tensor algebra in order to define signature (as suggested by the alternate name tensorial exponential map).

Definition 1.2 (Tensor Algebra of Formal Series)**.** For $n \in \mathbb{N}$, let $T^{(n)}(E) := \bigoplus_{k=0}^{n} E^{\otimes k} := \{a = (a_0, a_1, \dots, a_n) \,|\, \forall k \le n, a_k \in E^{\otimes k}\}$, i.e. the space of non-commutative polynomials of E-tensors of degree at most n, formally $\sum_{k=0}^{n} a_k x^{\otimes k}$ ($x^{\otimes k}$ is a placeholder for the k-th component), where $E^{\otimes 0} := \mathbb{R}$, the underlying scalar field by convention. Similarly, let $T^{(<\omega)}(E) := \bigoplus_{n=0}^{<\omega} E^{\otimes n} := \bigoplus_{n=0}^{\infty} E^{\otimes n} :=$

[1]The technical reason why we work in this space will be clear after seeing the definition of signature, Definition 1.3.

$\bigcup_{n\in\mathbb{N}}\bigoplus_{k=0}^{n} E^{\otimes k}$, i.e. the space of non-commutative polynomials of E-tensors, formally the same as above, with varying $n \in \mathbb{N}$. Finally, let $T^{(\omega)}(E) := \bigoplus_{n=0}^{\omega} E^{\otimes n} := \{a = (a_0, a_1, \dots, a_n, \dots) \,|\, \forall n \in \mathbb{N}, a_n \in E^{\otimes n}\}$, i.e. the space of non-commutative formal power series of E-tensors, formally $\sum_{n=0}^{\infty} a_n x^{\otimes n}$. All of $T^{(n)}(E)$, $T^{(<\omega)}(E)$, and $T^{(\omega)}(E)$ are non-commutative associative unital algebra over $\mathbb{R}$, with addition and scalar multiplication given componentwise, multiplication given by polynomial multiplication or sequence convolution (truncated at degree n for $T^{(n)}(E)$), the zero being sequence of graded zero tensors, and the unit being $1 = (1, 0, 0, \dots)$, scalar 1 followed by graded zero tensors. Denote the truncation at degree n by $\pi_n : T^{(m)}(E) \to T^{(n)}(E)$, which is a homomorphism, where $n \leq m \in (\omega + 1) = \mathbb{N} \cup \{\omega\}$ or m is $< \omega$. Note that the inclusion $T^{(<\omega)}(E) \hookrightarrow T^{(\omega)}(E)$ is also homomorphic, but it is not true if $< \omega$ is replaced by $n \in \mathbb{N}$.

For $n \in (\omega + 1)$, $a \in T^{(n)}(E)$ is invertible if and only if $a_0 \neq 0$. Let $\tilde{T}^{(n)}(E) := \{a \in T^{(n)}(E) \,|\, a_0 = 1\}$, the (special) group of elements with scalar term 1. Analogously, $\tilde{T}^{(<\omega)}(E)$ can be defined but it is not a group.

The algebraic properties of various spaces of tensors defined above will be further discussed in the next subsection. They will play a crucial role in studying paths, and in our case, justifying why signature is a reasonable candidate feature for handwriting recognition. Now, we are ready to give the definition of signature and truncated signature, the finite dimensional approximation in applications.

Definition 1.3 (Signature or Tensorial Exponential)**.** Let $X : I = [a, b] \to E$ be a continuous path of bounded variations. Define the signature or the tensorial exponential $S(X) \in T^{(\omega)}(E)$ of X by $S(X)_n := \int_{a<t_1<\cdots<t_n<b} \bigotimes_{i=1}^{n} \mathrm{d}X_{t_i} \in E^{\otimes n}, \forall n \in \mathbb{N}$, where $S(X)_0 := 1$ by convention. In the notation of formal series, we may write $S(X)(x) = 1 + \sum_{n=1}^{\infty} \int_{a<t_1<\cdots<t_n<b} \prod_{k=1}^{n} \mathrm{d}\langle X_{t_i}, x\rangle$. For $n \in \mathbb{N}$, the n-th truncated signature is given by $S^{(n)} := \pi_n \circ S$. The transform S ($S^{(n)}$) is called the (truncated) exponential map.

We remark that the iterated integral is componentwise well-defined, since $X \in C \cap BV(I; E)$, where $BV(I; E)$ is the topological dual of $C(I; E)$. Although a more general setup in rough path theory via Young's integral is possible, we stick with the better-understood space $C \cap BV$, in which more results are available[2].

One might wonder where the definition of signature via iterated integrals comes from. Indeed, the idea arises naturally in solving a linear ODE theoretically and numerically, as demonstrated below. The example of linear ODE somehow justifies the name exponential map as well (a more sophisticated connection is to Lie theory). Also, it is helpful to compare signature with Taylor series where iterated partial derivatives are taken locally around an expansion center a to obtain a (formal) power series (converging to the original function f in some open neighbourhood $U \ni a$ if and only if $f \in C^{\omega}$, i.e. analytic). We refer interested readers to Appendix A for a more elaborate discussion. While explaining the shuffle product formula at the end of this section, we shall provide another heuristic motivation to the definition of signature, with close connection to statistics, in particular linear regression.

Consider a linear differential equation driven by $X : I = [0, T] \to E$, that is $\mathrm{d}Y_t = A(\mathrm{d}X_t, Y_t), Y_0 = y_0 \in E$, where $A \in \mathcal{L}(E \times E; E)$, or a $(1, 2)$-E-tensor. It is well-known that the solution to this linear ODE uniquely exists: one applies the Picard-Lindelof iteration and shows its (eventual) contractivity. The iteration is actually a practical numerical scheme to compute the solution, starting with

[2]Young's integral is defined for continuous integrators of finite p-variations and continuous integrands of fintie q-variations, where $p, q \in [1, \infty)$ and $1/p + 1/q > 1$. One undesirable consequence of Young's integral is that Fubini's theorem does not hold when $p > 1$.

$Y_\cdot^{(0)} :\equiv y_0$, and inductively integrating $Y_0^{(n)} + \int_0^\cdot A(\mathrm{d}X_t, Y_t^{(n)}) =: Y_\cdot^{(n+1)}$. The closed form expression of the n-th approximation is

$$\begin{aligned}
Y_\cdot^{(n)} &= y_0 + \sum_{k=1}^{n} \int_0^\cdot A\left(\mathrm{d}X_{t_k}, \int_0^{t_k} A\left(\mathrm{d}X_{t_{k-1}}, \cdots, \int_0^{t_2} A(\mathrm{d}X_{t_1}, y_0)\right)\right) = y_0 + \sum_{k=1}^{n} \left(A\int_0^\cdot \mathrm{d}X_{t_k}\right)\left(A\int_0^{t_k} \mathrm{d}X_{t_{k-1}}\right)\cdots\left(A\int_0^{t_2} \mathrm{d}X_{t_1}\right) y_0 \\
&= y_0 + \sum_{k=1}^{n} A^{\otimes k}\left(\left(\bigotimes_{i=1}^{k-1}\int_0^{t_{i+1}} \mathrm{d}X_{t_i}\right)\otimes\int_0^\cdot \mathrm{d}X_{t_k}\right) y_0 = \left(I + \sum_{k=1}^{n} A^{\otimes k}\int_{0<t_1<\cdots<t_k<\cdot}\bigotimes_{i=1}^{k} \mathrm{d}X_{t_i}\right) y_0.
\end{aligned}$$

Iterated integrals appear inductively in the approximate solutions to the linear ODE, and observe that the operator $y_0 \mapsto Y_t^{(n)}$ is nothing but evaluating the formal tensorial polynomial $S^{(n)}(X|_{[0,t]})$ at the tensor A. As the iteration scheme converges to the solution, we have that $Y_t = \left(I + \sum_{k=1}^{\infty} A^{\otimes k}\int_{0<t_1<\cdots<t_k<t}\bigotimes_{i=1}^{k}\mathrm{d}X_{t_i}\right) y_0 = S(X|_{[0,t]})(A)(y_0)$, more than just an artificial formal construct in algebra. Moreover, as in the proof of Picard-Lindelof, one can show factorial decay of the summands, i.e. $\forall k \in \mathbb{N}_+$, $\left|\int_{0<t_1<\cdots<t_k<T}\bigotimes_{i=1}^{k}\mathrm{d}X_{t_i}\right| \le \frac{1}{k!}\|X\|_{BV([0,T];E)}^k$. The rate of convergence is exactly that in the MacLaurin series of exponential function. In the special case $d = 1$ (i.e.$E = \mathbb{R}$), the series of iterated integrals reduces to $Y_\cdot = \exp\left(A\int_0^\cdot \mathrm{d}X_t\right) y_0 = \exp(A(X_\cdot - X_0))y_0$ as usual. Note that it is not required that Y_t lives in the same space as X_t does, as long as A acts properly. If $X_t \in \mathbb{R}$ and $Y_t \in E$, then A can be represented as a square matrix and the solution to the ODE is still as above with matrix exponential.

One might find the tensorial algebraic exercise an overkill, but it turns out useful and (sometimes surprisingly) efficient in numerical computations. Suppose that we have to solve the ODE for many different infinitesimal generators[3] A and initial conditions y_0 (the initial conditions don't have to be in the same space as aforementioned!) but a fixed driving path X. Rather than performing Picard-Lindelof as above for every pair (A, y_0), we first compute and store the (truncated) signature of X independently, then evaluate the polynomial (the degree of the polynomial depends on the norm of A and tolerance for error) at different tensor A, which could be implemented efficiently. Essentially we are treating the path X as a constant but the pair of generator and initial condition (A, y_0) as the only variable here, a technique known as currying in computer science. Tensor algebra simply realizes this idea mathematically, separating (A, y_0) from X in the seemingly clumsy iteration. In this sense, the signature $S(X)$ summarizes all effects of linear operators on the path X, a fact comparable to Taylor series representation of an analytic function, and Hausdorff Moment Problem, whose solution confirms that the moments of a compactly supported random variable uniquely determines its probability distribution. It is now clear from the ODE example why iterated integrals are worth studying and why signature is also called the exponential map.

In the following subsection, we shall see that signature indeed contains "more" information: it is a faithful representation of the path up to a "small" set of "null" paths.

1.2 Tree-like Paths, Signature as Group Homomorphism and Faithful Representation

The goal of this subsection is to understand nice properties of signatures, culminating at the theorem in Hambly and Lyons [4] that the signature $S(X) \in T^{(\omega)}(E)$ is essentially a faithful representation of the path $X \in C \cap BV(I; E)$. Let's start from the basics.

[3]The solution to this ODE is obviously a time-homogeneous flow, and Lie theory kicks in its study. Signature then coincides with the exponential map in Lie group/algebra. In differential geometry, the ODE represents development of one manifold over another, in other words, rolling one surface onto another. Indeed, geometry is the primary origination of signature, in K.T. Chen's thesis around 60 years ago, per Hambly and Lyons [4].

Proposition 1.4 (Invariance under Translation and Time Change)**.** *Let $X \in C \cap BV(I;E)$ be a continuous path of bounded variations, $y \in E$ a vector, and $\tau : [a,b] =: I \to J := [c,d]$ a continuous and increasing function, then $X + y$ and $X_{\tau(\cdot)}$ are also continuous paths in E of bounded variations with the same signature as X, that is $S(X+y) = S(X) = S\left(X_{\tau(\cdot)}\right)$.*

The vector y is interpreted as translation while the function τ can be be viewed as time change. The invariance properties follow directly from those of (iterated) integrals. They will prove desirable in our mini-task of online handwriting digit recognition. Due to invariance, we sometimes abuse notation and identify paths differed only by a translation or a reparametrization[4]. Without loss of generality, we may always assume $I = [0,1]$, the unit compact interval, whenever convenient. We denote the path space $C \cap BV(I;E)$ under such identification by $Path(E)$.

The next thing to consider naturally is algebraic structure on $Path(E)$ and $T^{(\omega)}(E)$ that signature preserves. In contrast to Taylor series, the exponential map is not a linear operator, and hence vector space (and therefore algebra) is not the structure we are hunting for. Recall that $S(X)_0 \equiv 1$, $\forall X \in Path(E)$, which always lives in the group $\tilde{T}^{(\omega)}(E)$. So one should expect S being a (non-commutative) group homomorphism to $\tilde{T}^{(\omega)}(E)$. But what should be the corresponding group structure on $Path(E)$? Of course, we can always abstractly pullback the group structure $\left(\tilde{T}^{(\omega)}(E), \otimes, 1\right)$ to the set $Path(E)$, but it is more straightforward to guess (part of) the group structure: the multiplication is given by concatenation of paths, as in fundamental groups.

Definition 1.5 (Concatenation and Identity Path)**.** Let $X, Y \in Path(E)$ with respective representatives $X, Y \in C \cap BV(I;E)$ be two continuous paths of bounded variations, where $I := [0,1]$. Define the concatenated path $X * Y \in Path(E)$ via the representative in $C \cap BV(I;E)$ given by $(X*Y)_t := (X_{2t} - X_0)\mathbb{I}_{[0,1/2]}(t) + (Y_{2t-1} - Y_0)\mathbb{I}_{[1/2,1]}(t)$. Concatenation is well-defined, that is independent of the choice of representatives. It is associative (but not commutative) and hence makes the path space $Path(E)$ a semigroup, which is futhermore a non-commutative monoid, with the identity being the constant path $\mathrm{id} :\equiv 0$.

Proposition 1.6 (Signature as Monoid Homomorphism)**.** $S : (Path(E), *, \mathrm{id}) \to \left(\tilde{T}^{(\omega)}(E), \otimes, 1\right)$ *is a monoid homomorphism, that is $\forall X, Y \in Path(E)$, $S(X*Y) = S(X) \otimes S(Y)$ and $S(\mathrm{id}) = 1$.*

Preservation of identities is immediate as nontrivial integrals are zeros, while preservation of multiplications can be proved by partitioning the unit interval I into two halves and applying Fubini's theorem[5]. It seems that we are stating the obvious, but let's remark that this property allows us to compute the signature of a long path by "appending" signatures of many short ones, opening the door to optimize algorithms to large scale calculations or to perform real-time tasks as data streams feed in around the clock.

Now that $Path(E)$ is a monoid, can we push it to be a well-defined group by specifying the inversion? A first answer is no: the identity path is characterized by a singleton image, i.e. $X = \mathrm{id} \Leftrightarrow \#X(I) = 1$, and concatenation only enlarges images of paths. So other than the trivial case, X admits no inverse in $(Path(E), *, \mathrm{id})$, and we must impose extra structure on $Path(E)$ to make it a group. Mimicking the approach in fundamental groups, in which path reversal is the inversion, we shall identify $X * \overleftarrow{X}$ with id in $Path(E)$ and consider the equivalence relation "generated" by this identification, where $\overleftarrow{X}$ is the reversed path of X.

[4]Note that we do not identify paths with the same signature, at least have not yet. The "identification" is the content of the theorem on faithful representation by Hambly and Lyons.

[5]This is another technical reason why we restrict the paths to be continuous and of bounded variations.2

Definition 1.7 (Reversal and Tree-like Path)**.** Let $X \in Path(E)$ with representative $X \in C \cap BV(I = [0,1]; E)$ be a continuous path of bounded variations. Define $\overleftarrow{X} \in Path(E)$, the reversed path of X, via the representative $\overleftarrow{X} := X_{1-\cdot} \in C \cap BV(I; E)$. Reversal is well-defined, independent of representatives. Observe that double inversion is the identity function, i.e. $\overleftarrow{\overleftarrow{X}} = X$, and that inversion (reversely) respects concatenation, i.e. $\overleftarrow{Y} * \overleftarrow{X} = \overleftarrow{X * Y}$, where $Y \in Path(E)$.

X is called a tree-like path, if there exists a height function $h : [0,1] \to [0, \infty)$ such that $h(0) = 0 = h(1)$ and $\forall 0 \leq s \leq t \leq 1$, $\|X_t - X_s\| \leq h(s) + h(t) - 2 \inf h([s,t])$. Note that $X_0 = X_1$ is enforced in the definition of a tree-like path, and the height function is like a "group" version of the arc length function[6] of X. Indeed, the path $Y * X * \overleftarrow{Y}$ is tree-like provided X is, so is id trivially. Tree-like paths also form a sub-monoid of $Path(E)$ stable under reversal. Thus, the relation $X \sim Y$ defined by $X * \overleftarrow{Y}$ being tree-like is an equivalence relation preserved by concatenation and reversal, and $\left(RPath(E) := Path(E)/\sim, *, \mathrm{id}, \overleftarrow{\cdot}\right)$ is therefore a group.

The definition of tree-like paths may seem awkward at a first glance: why not simply use the equivalence relation generated by $X * \overleftarrow{X} \sim \mathrm{id}$? Well, as in many cases of algebra versus analysis, the subtlety is that the equivalence relation is finitely generated while tree-like paths covers the situation of (countably) infinitely many segments of the form $X * \overleftarrow{X}$. Nonetheless, it is conceptually very helpful to think of tree-like paths as the equivalence class of the identification $X * \overleftarrow{X} \sim \mathrm{id}$. In application where discrete approximations are used, this is particularly true: tree-like paths are finitely generated by such identification. In algebra and computer science, the group is better known respectively as the free group and the group of reduced words, presented as follows. Consider an $\mathbb{N}$-index path in a d-dimensional integral grid $\mathbb{Z}^d$. One can think of the positive unit vector in a distinct dimension as a distinct letter, and the negative one as its inverse. Then a path of finite length and unit speed in $\mathbb{Z}^d$ (such path must move along the grid and cannot jump across the diagonal etc.) can be represented by a word from the $2d$-letter alphabet consisting of the positive and negative unit vectors. If one treats "moving forward and immediately backward" as "not moving at all", i.e. $X * \overleftarrow{X} \sim \mathrm{id}$, then the paths on $\mathbb{Z}^d$ are exactly the reduced words from the d-letter alphabet consisting of the positive unit vectors. "Reduced" here means that the length of the words could be shortened by concatenating the inverse letter, like hitting the "backspace" dedicated for the letter on the keyboard. (Of course, if the "backspace" is for another letter, then the length of the word just increases by one. And the word "reduced" [7] is why we name the space $RPath(E)$.) The group of reduced words is the most popular construction of the concept of a free group of a finite set. Here we see that it also arises in the rough path theory when we are studying signatures, at least in the discrete case. From above, $RPath(E)$ can be viewed as a generalization of reduced words, and carries some flavor of the free group of paths in E.

Proposition 1.8 (Signature as Group Homomorphism)**.** $S : \left(RPath(E), *, \mathrm{id}, \overleftarrow{\cdot}\right) \to \left(\tilde{T}^{(\omega)}(E), \otimes, 1, \cdot^{-1}\right)$ *is a group homomorphism, that is* $\forall X, Y \in RPath(E)$*,* $S(X * Y) = S(X) \otimes S(Y)$*,* $S(\mathrm{id}) = 1$*, and* $S\left(\overleftarrow{X}\right) = S(X)^{-1}$.

A direct computation with some ODE theory shows that $S(X) \otimes S\left(\overleftarrow{X}\right) = 1$. One completes the proof by showing that signature respects tree-like paths, which is more technical. Interested readers could consult Hambly and Lyons [4] for the proofs of this fact and the following celebrated theorem, which says that exponential map is a lossless compression of all essential information of paths.

[6] One more technical reason why we restrict the paths to be continuous and of bounded variations.2
[7] NOT "rough"!

Theorem 1.9 (Signature as Faithful Representation). $S : \left(RPath(E), *, \mathrm{id}, \overleftarrow{\cdot}\right) \to \left(\tilde{T}^{(\omega)}(E), \otimes, 1, \cdot^{-1}\right)$ *is injective, i.e. the kernel of* $S : Path(E) \to \tilde{T}^{(\omega)}$ *is the set of tree-like paths. Moreover, among all paths in* $Path(E)$ *sharing the same signature, there is a unique one with the minimal arc length.*

We do warn readers that S is far from surjective, and the image of S can be further restricted to a proper Lie subgroup following the results in the next subsection and some classic Lie theory, though we are not going to pursue in this direction[8]. According to Geng [1], a characterization of the image of S is still unknown, even though the truncated counterpart is well understood.

At the end of this subsection, we emphasize again that the discrete sequence of signature encodes almost all information of a path in a graded fashion with increasing granularity, just like the (coefficients of) Taylor series to an analytic function, the moment sequence to a compactly supported random variable, or the spectra to a periodic function. Whence, studying paths with signatures is theoretically justified, and this is one of the two bedrocks in our mini-task in Section 2.

1.3 Shuffle Product Formula: Polynomial is Linear

So far we have been considering a path or its signature itself. In this section we shift gears to study functionals on the path and its signature. In other words, we treat a path as a black box and test its actions on functionals. It turns out surprisingly that linear functionals of signatures are all that we need to understand the underlying path, modulo the "null" segments of tree-like paths per Theorem 1.9. The magic here is the so-called shuffle algebra, the "dual" of the Lie algebra that $S(RPath(E))$ lives in[9]. Before formally introducing the shuffle product and the shuffle algebra, let's digress to an example to grasp the main idea.

Consider the following data analytic task: find a model of the weights of people given their sexes and heights, where a reasonably-sized sample dataset of the three variables are available. One of the models that many would try the first would be linear regression: $weight = \beta_0 + \beta_1 \cdot sex + \beta_2 \cdot height + \epsilon$. However, linear regression does not stop at this simple model if it does not fit the data well: one can do more with *only* linear regression. For example, if there turns out to be a statistically significant difference in the β_2 coefficients of heights between males and females, one can try another linear regression: $weight = \beta_0 + \beta_1 \cdot sex + \beta_2 \cdot height + \beta_{1,2} \cdot sex \cdot height + \epsilon$, with $\beta_{1,2}$, commonly referred to as the coefficient of interaction by statisticians, capturing the aforementioned difference between sexes. With a bit more prior knowledge, such as the assumption that densities across different people of the same sex are almost the same, one can propose a more sophisticated model but still with linear regression: $weight = \sum_{n=0}^{3} \left(\beta_n \cdot height^n + \gamma_n \cdot sex \cdot height^n\right) + \epsilon$. Generalizing this technique of adding powers of explanatory variables, any polynomial functional relation of the data can be recovered by linear regression. Thanks to Stone-Weierstrass, polynomial, and hence linear regression is theoretically powerful enough for us to find a desired model[10].

[8] In this case, one will dance among multiple algebras, including a Lie algebra of polynomials and series in which tensor product $\otimes$ is replaced with Lie bracket $[,]$. Exponential map on the Lie algebra will be distinguished from S and utilized to bound the set of signatures.

[9] A universal enveloping Lie algebra.

[10] Statistics, or data analytics, is sometimes more of an art than a science. We are reminding readers of the simple yet powerful tool of linear regression, long overlooked in a world filled with fancy machine learning, rather than advocating the superiority of linear regression over other methods. Fans of neuron networks should also be aware that the power of neuron networks comes from its own version of Stone-Weierstrass, the Universal Approximation Theorem.

Then what does linear regression have to do with paths and signatures? Intuitively, one can think of iterated integrals as "powers of paths", and that continuous functionals of paths can be approximated (under some suitable metric) by "polynomials of paths", which are "linear regression models of (truncated) signatures" in disguise. By moving from $C \cap BV(I; E)$ to a more sophisticated space $T^{(\omega)}(E)$, functionals of paths can be "linearized" to linear functionals of signatures of paths.The advantage is thus obvious: linear functionals are very well understood, both in theory and in numerics. Mathematicians would rather work with "bad objects in a good space" than with "good objects in a bad space"[11]. Echoing the first subsection, we remark that linearization to work in a "good space" could be considered another motivation to study iterated integrals of paths.

The idea of linearization of polynomials is clear from the example above, but what exactly is the magical space in which every polynomial is linear, a seemingly false statement especially confusing for most analysts? We devote the rest of this subsection to formalize the idea by introducing the shuffle product and the shuffle algebra. The key, the shuffle product formula, will justifies our restrictions to linear functionals of signatures when studying paths, the second bedrock in our mini-task in Section 2.

Let E^* be the dual of E[12], then it is clearly that $\left(T^{(n)}(E)\right)^* \cong T^{(n)}(E^*)$. Whence, we have the following linear embedding $T^{(<\omega)}(E^*) \hookrightarrow \left(T^{(\omega)}(E)\right)^*$. Take e^* and f^* in $T^{(<\omega)}(E^*)$, then we have two reals $e^*(S(X))$ and $f^*(S(X))$ for some fixed path $X \in RPath(E)$. To show that polynomials are linear, it suffices to find another $g^* \in T^{(<\omega)}(E^*)$ such that $g^*(S(X)) = e^*(S(X))f^*((S(X))$, and the rest is simply an induction argument. This is the content of the following theorem.

Theorem 1.10 (Shuffle Product and Shuffle Algebra)**.** *For every $e^*, f^* \in T^{(<\omega)}(E^*)$, there is a unique linear form, called the shuffle product of e^* and f^*, denoted by $e^* \sqcup f^* \in T^{(<\omega)}(E^*)$, such that $(e^* \sqcup f^*) \circ S = (e^* \circ S) \cdot (f^* \circ S)$ on $RPath(E)$. Therefore, the space $T^{(<\omega)}(E^*)$ restricted to $S(RPath(E))$ equipped with the shuffle product $\sqcup$ forms a commutative algebra, with the zero and unit the same as those with the tensor product $\otimes$.*

The theorem is rather abstract, but one can prove it constructively via the following combinatorial identity. The idea is to only consider e^* and f^* of the elementary form due to linearity. In this case, e^* and f^* are characterized by the strings or words of their components. The shuffle product formula is then a direct consequence of Fubini's theorem.

Definition 1.11 (Shuffle Product and Algebra of Words, Shuffle Permutations)**.** Let A be a d-letter alphabet, a and b be two words from the alphabet A of length m and n in $\mathbb{N}$, respectively. Define $a \sqcup b$, the shuffle product of a and b, to be the formal sum of all words of length $m + n$ interleaving both words a and b. Inductively, the shuffle product $\sqcup$ is defined by the following axioms:

- $a \sqcup \emptyset = a = \emptyset \sqcup a$;
- $(ax) \sqcup (by) = (a \sqcup (by))x + (b \sqcup (ax))y$, where $x, y \in A$.

A permutation $\sigma \in S_{m+n}$, the symmetric group of order $m + n$, is called a shuffle of m and n, denoted by $\sigma \in \text{Shuffle}(m, n)$, if $\forall 0 < i < m - 1, \sigma(i) < \sigma(i + 1)$, and $\forall 0 < j < n - 1, \sigma(m + j) < \sigma(m + j + 1)$. Then $a \sqcup b = \sum_{\sigma \in \text{Shuffle}(m,n)} c \circ \sigma$, where $c_i := a_i$ and $c_{m+j} := b_j$ for $1 \leq i \leq m$ and $1 \leq j \leq n$.

The free Abelian group of A-words equipped with the shuffle product $\sqcup$ (extended by linearity) thus forms a commutative algebra, with the unit being the empty word.

[11]The author learn this cute phrase from Antoine Remond-Tiedrez in his presentation of the project for the same course.
[12]Recall that E is finite dimensional, so the topology on E does not matter.

The name "shuffle" is indeed quite indicative: one can imagine shuffling a deck of m cards and another of n cards into a larger one of $m+n$ cards while preserving the orders in each original deck. There are $\binom{m+n}{m}$ such shuffles, each corresponding to a formal summand in the shuffle product $a \sqcup b$ before collecting the same terms. For example, $(xy) \sqcup z = xyz + xzy + zxy$, and $(xx) \sqcup (xxx) = 10xxxxx$ since $\binom{2+3}{2} = 10$, where $x, y, z \in A$.

We are now ready to state the shuffle product formula by specifying the following notations. Assume that $\dim E = d$, and set $A := [d] := \{n \in \mathbb{N}_+ \mid n \leq d\}$. Denote the dual basis vector of the a-th component by $\varepsilon_a \in T^{(<\omega)}(E^*)$, i.e. $\varepsilon_a(x) := x_a, \forall x \in T^{(\omega)}(E)$, where $a \in A$ is an index. We extend the notation by linearity if a is a formal sum of A-words.

Proposition 1.12 (Shuffle Product Formula)**.** *For A-words a and b,* $\varepsilon_a(S(X))\varepsilon_b(S(X)) = \varepsilon_{a \sqcup b}(S(X)) = \sum_{\sigma \in \mathrm{Shuffle}(m,n)} \varepsilon_{(ab)\circ\sigma}(S(X))$ *for every* $X \in RPath(E)$*. Therefore,* $\varepsilon_a \sqcup \varepsilon_b = \varepsilon_{a \sqcup b} = \sum_{\sigma \in \mathrm{Shuffle}(m,n)} \varepsilon_{(ab)\circ\sigma} \in T^{(<\omega)}(E^*)$.

The concept of shuffle product makes precise the idea of linearizing polynomials as presented in the example of linear regression. The shuffle product formula then asserts its feasibility on the range of signatures of paths. The message is that linear functionals of signatures form an algebra. In mathematics, algebras such as those of polynomials and of cylindrical functions are often considered nice in approximations. Here we have something better: linear functionals on signatures are enough to study paths, in particular to separate paths. In Section 2, we need to recognize handwriting digits, essentially requiring one to use some functionals to separate points in a space of features derived from recorded data. Based on discussions in this section, the partnership of signatures and linear functionals looks like a promising candidate.

2 Application of Signature to Online Handwriting Recognition

It is a platitude that pure mathematics can have unexpected consequences and affect even daily life.[13] As promised, we will apply the theory introduced in the previous section to perform a mini-task of handwriting digit recognition. Most readers may first find the application of signature to handwriting recognition unexpected[14]: what would be the (natural) paths to work on? It seems more straightforward if we were to study financial time series or soundtracks with signatures, in which case the "paths" are obvious. Well, although there have been applications to financial time series in Gergely Gyurkó et al. [2], Lyons et al. [10] and soundtracks according to Graham [3], the author thinks that both financial time series and soundtracks have been long well analyzed by other more traditional techniques and that each has its own system of terminologies possibly arcane to novice, while handwriting recognition is a relatively new and fast-developing field easier for beginners to catch up. Moreover, high-quality and small-sized data are freely available online, while most well-organized financial data costs money and soundtracks are huge in size. But most importantly, the author was intrigued by Geng's comment in [1] that signature-based handwriting recognization is under commercialization and Graham's victory in the Chinese character recognition competition of the Twelfth International Conference on Document Analysis and Recognition (ICDAR 2013) with a test error of 3.58% [3]. We will familiarize ourselves of relevant jargons in handwriting recognition in the first subsection before listing the pros and cons of signatures as extracted features in the second.

[13]Quote of the great British Mathematician Littlewood.
[14]This was exactly the reaction when Geng mentioned Chinese handwriting recognition in his presentation [1]

The last subsection reports the data, the algorithms and work flows, and the results of the mini-task of recognizing handwriting digits, from "0" to "9". Unlike Graham's winning algorithm exploiting the state of the art Convolutional Neuron Networks (CNN) [3], we will stick with linear functionals as Section 1 points out.

2.1 Handwriting Recognition: Online VS Offline

Essentially, in this subsection we are answering the question what the (natural) paths are in signature-based handwriting recognition. Depending on whether the source data contain the ordering information of handwriting or not, handwriting recognition falls into either category of online recognition and offline recognition. In online recognition, the pixels written on the panel (possibly along with other information such as pressures on the panel and angles of the stylus) are recorded in chronological order by a fixed sampling frequency, and this ordering of pixels are preserved in the list in the dataset available for recognition. In contrast, the dataset of offline recognition only contains the pixels that have been written on regardless of their orders, essentially treating handwritings as two-dimensional pictures or snapshots. Clearly, online dataset contains the extra structure of total order which can be "forgotten" to "recover" offline dataset. A more mathematical point of view is that online dataset is a one-parameter (time) representation in the two-dimensional panel whereas offline dataset is not. In other words, online data are naturally paths per our definition in Section 1, and hence one can study online handwriting with signatures.

A majority of handwriting recognition tasks are to identify the characters written. Another aspect of handwriting recognition is to identify the writers. So handwriting recognition can also be categorized as writer-independent and writer-dependent. Equipped with the extra order information, online dataset gives a significant edge over offline dataset in writer-dependent recognitions. Indeed, there are studies of converting an offline handwriting to the most likely online one, for the purpose of identifying writers. In writer-independent tasks, the order information may occasionally mislead some online recognition algorithms while offline recognitions are automatically blind to the directions and the orders of how different people write down the same characters. Even though we are performing only online writer-independent recognition in the mini-task, we had better be aware of the categorization in terms of writer and its relation to online/offline categorization.

It is worth pointing out that when speaking of handwriting recognition, most people would actually think of offline writer-independent recognition. Quite a lot of work has been devoted to offline writer-independent recognition and achieved amazing results during the past two decades, with the dataset MNIST[6] being the most popular one to benchmark performance. However, since MNIST is an offline dataset which does not contain a natural path structure, we have to find another online dataset for our purpose. To the best of the author's knowledge, there has not yet been an online counterpart of MNIST as the gold standard dataset. For simplicity, we shall use the free dataset archived by University of California Irvine (UCI) Machine Learning Repository, with further details explained in the last subsection.

2.2 Signature as Extracted Features: Apriori Analysis

Before playing with the actual data, let's develop some qualitative heuristics of the feature space. During this brainstorming process, we can identify some necessary preprocessings and postprocessings while avoiding silly operations. It also helps with debugging or further investigation in case we encounter unexpected results.

Recall that signature is invariant under translation and reparametrization, which is ideal for handwriting recognition: we don't want to distinguish two identical digits written in different positions of the panel or recorded in different time frames. Unlike most other algorithms which require centering in preprocessing, we can skip it when using signature as features. Unfortunately, signature is sensitive to scaling and rotation. One probably does not want to learn the sizes of handwritings[15], so normalization would be a great preprocessing technique to try in our experiment. Since we are studying paths, we should normalize them to the same length, rather than the same area as in some offline recognition algorithms. As for rotations, rotating by a small angle might help, but one has to figure out the optimal rotation angle or some synthesis algorithm combining different rotated handwritings. The idea of bootstrapping from perturbed handwritings seems promising, and we will treat rotation as a special case of noise of perturbation. But bootstrapping and randomization is more than just a simple preprocessing, and we shall leave it for future studies.

Since signature is nothing but a sequence of definite integrals, it captures the global geometric features rather than the local properties of a path. For example, the first order integral (as a d-dimensional vector) is precisely the position of ending point relative to the starting point, while the second order integrals are related to the areas enclosed by the path and some straight line segments. Signature is not obsessed to the slopes of the path at some sampled points as some gradient-based recognition algorithms, but looks at the big picture of the entire character, which seems closer to the way human beings recognize handwritings. Numerically, iterated integrals are not easy to compute and the number of components grows exponentially as the order (the number of iterations) grows, heavily tolling the memory and storage. In reality, one has to determine a reasonable order to truncate the signature for the purpose of the tasks, striking a balance between performance and resources.

In introducing the concepts of online and offline recognitions, we mention that the total order structure may sometimes be undesirable in writer-independent recognitions. But this is only a minor issue compared to multiple connected components or pen strokes. A handwritten character is sometimes more than a single path: one lifts the pen off the panel and puts down the pen at another position in writing a single character such as "4" or "5". Jumps in the paths are not encoded in signatures as we restrict paths to be continuous. For example, a line segment and its two endpoints shall share the same signature. To overcome this significant drawback of signature, we can introduce a third dimension to record the jumps: the third component is zero whenever the stylus is touching the panel; and the component is one otherwise. This idea of extra dimension is inspired by the very fact that we live in a three-dimensional world and the conviction that "physical processes are continuous". Furthermore, if we have the pressure data available, then we may use pressure as the third dimension. Thus, every handwriting regardless of the number of pen strokes can be represented as a continuous path in a three-dimensional space. We shall append the third dimension of jump indicator when reading the raw data. However, one should also keep in mind that the extra dimension is quite redundant in memory and storage.

[15] Unless one tries to accomplish the almost impossible task of telling apart "o" and "O". Well, relative sizes do matter in a sentence, but one can also use other information such as grammar in a sentence to tell them apart.

To sum up, signature-based recognition is invariant under translation and reparametrization, and insensititve to local noises of handwritings, but one has to address issues with scaling, rotation, jumps, total order structure, and computational efficiency in terms of both time and space. Some preprocessing techniques such as normalization are available to mitigate the shortcomings.

2.3 Experiment: Recognizing Digits using Signature and Linear Functionals

In this last subsection, we shall describe our data source and work flow, and report the results of our mini-task. MATLAB [12] along with its official Statistical and Machine Learning Toolbox is used throughout the study. All files including illustrations are available on GitHub at `https://github.com/yangxikevinou/DigitSignature`[16].

2.3.1 Data Source and Description

The online handwritten digits dataset is downloaded from University of California Irvine (UCI) Machine Learning Repository [8]. According to the Repository, the dataset contains samples from 44 writers, each contributing 250 handwritten digits from "0" to "9" not necessarily in particular orders. It has already been divided into a training dataset with 30 writers and a test dataset with 14. The actual training set has 7494 samples (six fewer than the claimed 7500) while the actual test set has 3498 (two fewer than the claimed 3500), with a total size of less than 1.5 megabytes combined. The sample size is large enough for us to reach statistically meaningful conclusions with appropriate methods. Since we are conducting writer-independent recognitions, we do not mind the writer labels and the missing entries in the sample[17].

The raw data are in the so-called UNIPEN format. Each handwriting begins with a header line indicating the digit being written. Pixels on a 500×500 panel recorded at a rate of 10 Hertz are presented line-by-line in tab-delimited format, with the first column being the horizontal coordinate and the second vertical. Each pen stroke is encapsulated between a line ".PEN_DOWN" and a line ".PEN_UP". We refer interested readers to the UCI webpage [8] for more details regarding the source data, including the hardware and the collection process.

There is also a smaller online dataset in the same format containing all 36 alphanumerals repeated twice from each of the 11 writers available in the UCI repository [7]. The author used this dataset in the development, too.

2.3.2 Program Design and Work Flow

There are four major steps in the experiment, the third of which is the most important. We sketch the work flow step-by-step as follows. All m-files appearing below can be found in `https://github.com/yangxikevinou/DigitSignature`.

1. **Reading raw data**

   The function "readpen.m" reads raw data from the text files in UNIPEN format into a dataset array in MATLAB, in which the

[16] Interested readers should contact the author first for use outside of MATH 820 of Fall 2016 at Carnegie Mellon University.
[17] Those entries are missing in the available raw data, not caused by the author's reading and cleanup process.

first column is the label of digit in character data type and the second column is the path in three dimensional space stored in a cell array (to accommodate different numbers of sampled pixels in handwritings). In a single pen stroke, the two-dimensional pixels are appended by zeros in the third dimension. The line of ".PEN_DOWN" are replaced by the next pixel with one in the third dimension, while the line of ".PEN_UP" are replaced by the previous pixel with one in the third dimension. A three-dimensional augmented pixel is stored as a column vector, so a handwriting is a $3 \times n$ matrix.

MATLAB built-in functions for importing data and processing strings are used for efficiency. This step is executed only once since we stick with the same dataset. If another data source is used, then the function has to be modified if not overhauled.

2. **Visualizing handwritings**

   The function "writedigit.m" takes a matrix representing an online handwriting as input, animates the handwriting in chronological order, and outputs a movie frame structure for playback. The function scales and centers the handwriting on the plane, and indicates the position of ".PEN_DOWN" with a green downward pointing equilateral triangle and the position of ".PEN_UP" with a red upward pointing counterpart. The time interval between consecutive sampled points in the animation can be set as an optional input.

   MATLAB animated plot structure is used in implementation. One can further use image processing functions to create .gif files from the output movie frames for the purpose of presentation. This step is technically unnecessary in computing signatures or recognizing handwritings. However, it gives an intuition of the digitized data, and allows researchers to pay attention to possible pathologies or outliers.

   We remark that a naive plot of the two-dimensional pixels does not faithfully visualize the original offline handwriting, because MATLAB by default scales the horizontal axis and vertical axis differently. A very tall "1" would be shortened, while a fat "0" would be squeezed.

3. **Normalizing paths and computing signatures**

   The function "npath.m" takes a matrix representing an online handwriting and the length of the handwriting as inputs, and outputs a matrix of the same dimension representing the online handwriting with the length normalized to the given input. The length of a handwriting is the total variations of the continuous part of the path on the panel. The extra dimension (indicator of jumps or pressure) is not normalized.

   Then function "sig.m" takes a matrix of d rows representing an online handwriting and the order of truncation n as inputs, and outputs a row vector of dimension $1 \times \sum_{k=0}^{n} d^k$ representing the signature of the handwriting truncated at order n. The first component is always 1 as shown in the first section. The components of the k-th order iterated integrals are computed inductively from the $(k-1)$-th ones, leveraging on MATLAB built-in functions "kron". Integrations are approximated using the built-in function "cumsum". Iterated integrals of the same order are listed lexicographically, changing from the inner layers to the outer layers. For example, if the input matrix has two rows, i.e. the handwriting data is two-dimensional, then the third order iterated integrals are listed as "111, 112, 121, 122, 211, 212, 221, 222", where "*abc*" means that one integrates with respect to Coordinate c first, then Coordinate b, and lastly Coordiate a. Readers can refer to Appendix B for the source code.

We apply the function “npath” first to normalize the path first, then compute its truncated signatures using “sig” up to a certain order. With MATLAB built-in function “cellfun”, we apply them to an entire dataset array to obtain a matrix of features, with each row corresponding to the truncated signature of a handwriting and each column corresponds to a component (an iterated integral) of truncated signatures.

4. **Classifying handwritings with linear functionals**

   We use the truncated signatures of both training and test datasets computed in Step 3 as features to classify handwritings in test datasets into one of the ten digits. The MATLAB built-in function “classify” is used, whose default classifier is a linear classifier with multivariate normal densities and joint covariances. In the simplest case in which there are only two groups to distinguish, “classify” produces nothing but an optimal separating hyperplane assuming that data points follow normal distributions in each group, the so-called Support Vector Machine (SVM) in the (seemingly fancy) language of machine learning. In the case of multiple groups, “classify” separates each pair of groups with a calibrated hyperplane, creating a Voronoi diagram as a classifier. If we input the signatures of the test dataset, the signatures of the training dataset and the corresponding digits, then “classify” outputs the digits recognized by the signature-based linear classifiers.

2.3.3 Results

We try to run the classifier on three-dimensional online data with the third one being the jump indicator. Due to degeneracies in the truncated signatures, “classify” returns an error of zero variances/covariances. Thus, we only run the classifier on two-dimensional data without telling the jumps apart, and identifies a reasonable order of truncation and studies the misclassification error and further the confusion matrix.

We compute the signatures truncated at the 10th order, totalling 2047 components. Note that the first component is always zero and hence not informative, so we actually have 2046 features to play with. Since the signatures are computed inductively, a rigorous feature selection is neither useful nor practical. Instead, we shall compare the misclassification errors of different orders of truncations, and plot its confusion matrix, as follows, which can be produced by running the last for-loop in “demo.m”.

```
'order'    'resub error'    'test error'
[   10]    [     0.0017]    [    0.0992]


'true\pred'    [  0]    [  1]    [  2]    [  3]    [  4]    [  5]    [  6]    [  7]    [  8]    [  9]
[         0]    [335]    [  0]    [  0]    [  1]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]
[         1]    [  2]    [301]    [ 11]    [  1]    [  4]    [  4]    [  1]    [  1]    [ 10]    [  1]
[         2]    [ 24]    [  7]    [316]    [  6]    [  0]    [  0]    [  7]    [  0]    [  2]    [  2]
[         3]    [ 13]    [  0]    [  1]    [343]    [  1]    [  0]    [  1]    [  3]    [  0]    [  2]
[         4]    [  2]    [ 11]    [ 23]    [  0]    [298]    [ 14]    [  4]    [  3]    [  6]    [  3]
[         5]    [ 19]    [  0]    [  4]    [  1]    [  0]    [334]    [  0]    [  0]    [  0]    [  5]
[          6]    [  1]    [  3]    [ 11]    [  0]    [  4]    [  3]    [329]    [  0]    [  0]    [ 13]
[          7]    [  4]    [ 17]    [  3]    [  3]    [ 10]    [  5]    [  4]    [267]    [ 16]    [  6]
[          8]    [ 13]    [  1]    [  6]    [  0]    [  1]    [  0]    [  2]    [  1]    [312]    [  0]
[          9]    [  5]    [  0]    [  4]    [  6]    [  0]    [  2]    [  0]    [  2]    [  1]    [316]

'order'    'resub error'    'test error'
[    6]    [     0.0464]    [    0.0849]

'true\pred'    [  0]    [  1]    [  2]    [  3]    [  4]    [  5]    [  6]    [  7]    [  8]    [  9]
[          0]    [335]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  1]
[          1]    [  0]    [330]    [  2]    [  0]    [  1]    [  0]    [  0]    [  1]    [  2]    [  0]
[          2]    [  0]    [  1]    [253]    [  7]    [  0]    [  3]    [100]    [  0]    [  0]    [  0]
[          3]    [  2]    [  2]    [  3]    [340]    [  1]    [  0]    [ 16]    [  0]    [  0]    [  0]
[          4]    [  0]    [  0]    [ 54]    [  0]    [301]    [  0]    [  7]    [  0]    [  2]    [  0]
[          5]    [  0]    [  0]    [  0]    [  0]    [  0]    [363]    [  0]    [  0]    [  0]    [  0]
[          6]    [  1]    [  2]    [ 12]    [  0]    [  0]    [  0]    [348]    [  1]    [  0]    [  0]
[          7]    [  0]    [ 43]    [  1]    [  0]    [  4]    [  0]    [  0]    [283]    [  3]    [  1]
[          8]    [  0]    [  2]    [  8]    [  0]    [  1]    [  0]    [  0]    [  1]    [323]    [  1]
[          9]    [  0]    [  7]    [  2]    [  1]    [  0]    [  1]    [  0]    [  0]    [  0]    [325]

'order'    'resub error'    'test error'
[    5]    [     0.0677]    [    0.0995]

'true\pred'    [  0]    [  1]    [  2]    [  3]    [  4]    [  5]    [  6]    [  7]    [  8]    [  9]
[          0]    [335]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  1]
[          1]    [  0]    [306]    [ 11]    [  4]    [  8]    [  0]    [  0]    [  1]    [  6]    [  0]
[          2]    [  8]    [  0]    [238]    [  6]    [  6]    [  0]    [106]    [  0]    [  0]    [  0]
[          3]    [  0]    [  9]    [  1]    [354]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]
[          4]    [  0]    [  0]    [ 53]    [  0]    [305]    [  0]    [  2]    [  0]    [  4]    [  0]
[          5]    [ 10]    [  0]    [  0]    [  0]    [  0]    [353]    [  0]    [  0]    [  0]    [  0]
[          6]    [  0]    [  0]    [ 31]    [  0]    [  0]    [  0]    [332]    [  1]    [  0]    [  0]
[          7]    [  0]    [ 42]    [  4]    [  0]    [  2]    [  0]    [  1]    [279]    [  7]    [  0]
[          8]    [  3]    [  2]    [  4]    [  2]    [  0]    [  0]    [  0]    [  2]    [323]    [  0]
[          9]    [  0]    [  7]    [  0]    [  3]    [  0]    [  1]    [  0]    [  0]    [  0]    [325]
```

```
'order'    'resub error'    'test error'
[    4]    [     0.0973]    [    0.1286]

'true\pred'    [  0]    [  1]    [  2]    [  3]    [  4]    [  5]    [  6]    [  7]    [  8]    [  9]
[         0]    [334]    [  0]    [  1]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  1]
[         1]    [  0]    [292]    [ 19]    [  3]    [ 16]    [  0]    [  0]    [  1]    [  4]    [  1]
[         2]    [ 13]    [  0]    [210]    [ 16]    [  2]    [  0]    [123]    [  0]    [  0]    [  0]
[         3]    [  0]    [  4]    [  0]    [357]    [  0]    [  0]    [  0]    [  0]    [  1]    [  2]
[         4]    [  0]    [  0]    [ 55]    [  0]    [281]    [  0]    [  4]    [  0]    [ 24]    [  0]
[         5]    [  0]    [  0]    [  0]    [  0]    [  0]    [363]    [  0]    [  0]    [  0]    [  0]
[         6]    [  1]    [  1]    [ 30]    [  0]    [  3]    [  0]    [326]    [  3]    [  0]    [  0]
[         7]    [  0]    [ 71]    [  1]    [  0]    [  2]    [  0]    [  0]    [244]    [ 16]    [  1]
[         8]    [  2]    [  2]    [  3]    [  0]    [ 10]    [  0]    [  0]    [  1]    [318]    [  0]
[         9]    [  1]    [  9]    [  0]    [  2]    [  0]    [  1]    [  0]    [  0]    [  0]    [323]

'order'    'resub error'    'test error'
[    3]    [     0.1390]    [    0.1572]

'true\pred'    [  0]    [  1]    [  2]    [  3]    [  4]    [  5]    [  6]    [  7]    [  8]    [  9]
[         0]    [330]    [  5]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  0]    [  1]
[         1]    [  0]    [267]    [ 29]    [  5]    [ 10]    [  0]    [  0]    [  1]    [ 24]    [  0]
[         2]    [  0]    [  1]    [221]    [ 12]    [  2]    [  0]    [123]    [  0]    [  0]    [  5]
[         3]    [  2]    [  4]    [  0]    [345]    [  0]    [  0]    [  0]    [  0]    [  0]    [ 13]
[         4]    [  0]    [  2]    [ 69]    [  0]    [260]    [  0]    [ 12]    [  0]    [ 21]    [  0]
[         5]    [ 19]    [  0]    [  0]    [  0]    [  0]    [343]    [  0]    [  0]    [  0]    [  1]
[         6]    [  1]    [  0]    [ 36]    [  2]    [  5]    [  0]    [320]    [  0]    [  0]    [  0]
[         7]    [  0]    [ 63]    [  0]    [  0]    [ 18]    [  0]    [  1]    [214]    [ 22]    [ 17]
[         8]    [  0]    [  0]    [  6]    [  0]    [  7]    [  0]    [  0]    [  1]    [322]    [  0]
[         9]    [  1]    [  6]    [  1]    [  1]    [  0]    [  0]    [  0]    [  1]    [  0]    [326]
```

We are able to achieve around 90% accuracy or 10% misclassification error with order of truncation equal to 6. The classification runs in almost real time, even on the author's 7-year-old laptop with Intel Core 2 Duo 32-bit processor and MATLAB R2009a. The only preprocessing is normalization, and we have not yet factored jumps into the classifier.

3 Summary

In this report, we introduce the basic theory of signatures and apply it to a crude project of recognizing standalone handwriting digits. On the theory side, signature encodes almost all information of a continuous path of bounded variations in its profound algebraic structures of iterated integrals. One amazing consequence is the shuffle product formula, which turns every "polynomial" of the path into a linear functional of the signature. On the application side, online handwriting data can be treated as paths in spaces, and their signatures can be computed efficiently as extracted features in recognition tasks. Because of the shuffle product formula, using a "naive" linear classifier on signatures can be sufficient if properly executed. The result of our mini-task of handwriting digit recognition echoes existing research of signature-based methods in machine learning.

While the vast majority of research in machine learning discusses algorithms, it is worth keeping in mind the saying "garbage in garbage out". Features with nice (mathematical) properties such as signatures are also crucial to the outcomes and can sometimes reduce the complexities of algorithms needed. They deserve more attentions than they currently receive. For mathematicians, it would be a challenging yet interesting adventure to identify, formulate, and study these feature spaces, for example, the space of signatures.

Appendix A Signature and Taylor Series in Tensor Algebra

Tensor algebra is introduced before the definition of signature. Many might find it either exotic or nothing but only a bookkeeping at the very beginning to bundle iterated integrals in tensorial formal series. Well, quite the contrary, even undergraduates in mathematics have encountered a famous and significant example of tensorial formal series in disguise before the end of their second years, namely the Taylor series. We illustrate the point by rewriting Taylor's expansion in tensors.

Let $U \subseteq \mathbb{R}^d$ be open, $f \in C^\omega(U;\mathbb{R})$ and $a \in U$ with radius of convergence $r > 0$. Then by Taylor's theorem, on $B(a,r) \subseteq U$,

$$
\begin{aligned}
f(x) = \sum_{\alpha \in \mathbb{N}^d} \frac{1}{\alpha!}\frac{\partial^\alpha f}{\partial x^\alpha}(a)(x-a)^\alpha = \sum_{n=0}^{\infty}\frac{1}{n!}\sum_{|\alpha|=n}\frac{n!}{\alpha!}\frac{\partial^\alpha f}{\partial x^\alpha}(x-a)^\alpha = f(a) + \sum_{n=1}^{\infty}\frac{1}{n!}\sum_{\substack{1\le i_k \le d\\ 1\le k\le n}}\left(\frac{\partial}{\partial x_{i_n}}\cdots\frac{\partial}{\partial x_{i_1}}f\right)(a)\left((x-a)_{i_1}\cdots(x-a)_{i_n}\right) \\
= f(a) + \sum_{n=1}^{\infty}\frac{1}{n!}\left\langle\left(\left(\frac{\partial}{\partial x}\right)^{\otimes n} f\right)(a), (x-a)^{\otimes n}\right\rangle = \sum_{n=0}^{\infty}\frac{1}{n!}\left\langle\left(D^{\otimes n} f\right)(a), (x-a)^{\otimes n}\right\rangle =: (T_a f)(x-a),
\end{aligned}
$$

where $T_a f$ stands for the (formal) Taylor series of f around the center a, which is an element in the tensor algebra $T^{(\omega)}(\mathbb{R}^d)$ defined in the first section.

From this point of view, organizing iterated integrals as a formal series, namely signature, is a natural idea just as one deals with iterated derivatives in Taylor series. But even better, now that both concepts are expressed in the same notation, by Fundamental Theorem of Calculus, which asserts that integrations and differentiations are inverse operations to each other, one can pair them together and obtain a "duality" as follows.

Assume that $X \in C \cap BV([0,T];\mathbb{R}^d)$ starts at $X_0 = a$ and stays inside the ball $B(a,r) \subseteq U$ given above, then

$$
\begin{aligned}
f(X_T) = (T_a f)(X_T - a) &= \sum_{n=0}^{\infty} \frac{1}{n!} \left\langle \left(D^{\otimes n} f\right)(a), \left(\int_0^T \mathrm{d}X_t\right)^{\otimes n} \right\rangle = f(a) + \sum_{n=1}^{\infty} \frac{1}{n!} \left\langle \left(D^{\otimes n} f\right)(a), \int_{\substack{0<t_k<T \\ 1\le k\le n}} \bigotimes_{k=1}^{n} \mathrm{d}X_{t_k} \right\rangle \\
&= f(a) + \sum_{n=1}^{\infty} \left\langle \left(D^{\otimes n} f\right)(a), \frac{1}{n!} \sum_{\sigma\in S_n} \int_{0<t_1<\cdots<t_n<T} \bigotimes_{k=1}^{n} \mathrm{d}X_{t_{\sigma(k)}} \right\rangle \\
&= f(a) + \sum_{n=1}^{\infty} \left\langle \frac{1}{n!} \sum_{\sigma\in S_n} \left[\left(D^{\otimes n} f\right)_{i_{\sigma(1)},\dots,i_{\sigma(n)}}\right]_{\substack{1\le i_k\le d \\ 1\le k\le n}} (a), \int_{0<t_1<\cdots<t_n<T} \bigotimes_{k=1}^{n} \mathrm{d}X_{t_k} \right\rangle = f(a) + \sum_{n=1}^{\infty} \left\langle \left(D^{\otimes n} f\right)(a), S(X)_n \right\rangle \\
&= \left\langle S(X), \sum_{n=0}^{\infty} \left(D^{\otimes n} f\right)(a) \right\rangle = \left(S(X)(D)f\right)(X_0).
\end{aligned}
$$

This elegant "duality" relation somehow suggests that the seemingly artificial construct of signature should be valued as much as, if not more than, its counterpart, Taylor series.

Appendix B MATLAB Implementation of Signature Computation

We use MATLAB built-in function "kron" and "cumsum" to compute iterated integrals "vectorially" for efficiency. Since vectorization involves arrays of large dimensions growing exponentially, one might encounter memory outflows with high-dimensional paths or large orders of truncation. In this case, one should rewrite the code with loops and possibly writing large arrays into files, at the cost of speed.

The code snippet is imported via the M-code LATEXpackage [5].

```
function sig = sig(path,N)
% input
% path: dxn array
% N: natural number (may be zero), degree
% output
% sig: 1+d+d^2+...+d^N column vector, in each grade permutation of words in
% increasing order
%
% return the N-th degree truncated signature of path


% differential
path=diff(path,1,2);

% initialize
[d,D]=size(path);
pre=ones(1,D);
```

```
sig=1;
D=1;

for i=1:N
    pre=cumsum(repmat(pre,[d,1]).*kron(path,ones(D,1)),2);
    D=D*d;
    sig=[sig;pre(:,end)];
end

end
```

Acknowledgments

The author thanks Dr. Geng for introducing the fascinating and rapidly growing field of rough path theory in his presentation [1]. This is the first time that the author is aware of the theory of rough path, and in particular signatures. Dr. Geng also points the author to the compact references Lyons et al. [11, 10].

The author thanks Professor Slepcev for assigning the final project in his Fall 2016 offering of the course MATH 820 Advanced Topics in Analysis: Variational and PDE Techniques in Data Analysis, and teaching the author geometry in his Spring 2016 offering of the course MATH 759 Differential Geometry. It is the pressure from the course to actually "do something" that drives the author to adventure some extra miles into his unknown territory, but if not equipped with basic tools from geometry, the author might not survive in this foreign wonderland filled with fancy abstract nonsense. (Clive Newstead also helps indirectly by advertising the course PHIL 713 Category Theory.)

The author would also like to thank Antoine Remond-Tiedrez for informing of the powerful public computing resources on campus of Carnegie Mellon University, which make animated visualization and intensive model validations (and future to-do optimizations) much easier for the author (with a shitty 7-year-old 32-bit machine running MATLAB R2009a). The comment of "good space, bad objects" versus "bad space, good objects" comes from Antoine as well.

Bibliographic Notes

Section 1 is adapted from Chapter 2 in Lyons et al. [11], originating from a lecture note given at the (annual) Saint-Flour Probability Summer School in July 2004. The majority of the chapter is summarized in this report, with the exception of materials relating to Lie theory (and proofs!). Geng's presentation [1], mainly on the work of Hambly and Lyons [4], is also blended into this section. The lecture note by Lyons et al. [11] is rigorous enough whenever proofs are included, while the paper Hambly and Lyons [4] is a more serious treatment, and itself the proof of Theorem 1.9 (Signature as Faithful Representation). For more intuitions and a broader view, one can refer to the survey article Lyons [9] written for the International Congress of Mathematicians 2014, Korea, in which Hairer received his Fields medal .

Section 2 reports the design and findings of the mini-task of online handwriting digit recognition that the author conducts. The mini-task is inspired by a comment in Geng's presentation [1] that Chinese handwriting recognition based on signatures has been or is being commercialized. The paper by Graham [3] is one of the rare English literature that documents signature-based handwriting recognition. The author also casually learns the jargon and some practices of handwriting recognition from researchers' webpages, such as LeCun et al. [6], MATLAB [12, 13]. The raw data is downloaded from University of California Irvine (UCI) Machine Learning Repository [7, 8]. Although MNIST [6] is considered a benchmark dataset in handwriting recognition, it is offline and does not suit the situations here. Online data are fewer and harder to find than offline data. The author is responsible for the majority of contents in Section 2 and the MATLAB codes used, although MATLAB [12, 13] help files and support webpages are heavily referenced. For other analyses of data streams, references include the papers Lyons [9], Lyons et al. [10], Gergely Gyurkó et al. [2], where the latter two concentrate on applications to financial time series (which the author has better knowledge of and more experience with).

The Taylor series part of Appendix A is standard material in any undergraduate mathematical curriculum. The author rewrites it in the (alien) tensorial notations to make a point in formality and pair with signature aesthetically. The author has not yet encountered or been aware of literature in such notations to the best of his knowledge, and finds it absurd in the statement and usual applications of Taylor's theorem. This appendix was not originally intended, but inspired by Professor Slepcev's comment in the author's talk that one should consider Taylor series as a first example of formal tensor series.

Appendix B is included for completeness and to support the claim that signature can be computed efficiently, albeit a seemingly scary monster in mathematics and notations. The computation of signature is the most mathematical piece in the implementation of the handwriting recognition mini-task in this report.

References

[1] Xi Geng. The exponential transform of a path: a faithful representation. Center for Computational Finance (CCF) Seminar, 10 2016. URL `http://math.cmu.edu/CCF/ccfseminar.php?SeminarSelect=1331`.

[2] L. Gergely Gyurkó, T. Lyons, M. Kontkowski, and J. Field. Extracting information from the signature of a financial data stream. *ArXiv e-prints*, July 2013.

[3] Benjamin Graham. Sparse arrays of signatures for online character recognition. *CoRR*, abs/1308.0371, 2013. URL `http://arxiv.org/abs/1308.0371`.

[4] B. Hambly and T. Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *ArXiv Mathematics e-prints*, July 2005.

[5] Florian Knorn. M-code LaTeX Package. URL `http://www.mathworks.com/matlabcentral/fileexchange/8015-m-code-latex-package`.

[6] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits. URL `http://yann.lecun.com/exdb/mnist/index.html`.

[7] M. Lichman. UCI machine learning repository UJI pen characters data set, 2013. URL `http://archive.ics.uci.edu/ml/datasets/UJI+Pen+Characters`.

[8] M. Lichman. UCI machine learning repository pen-based recognition of handwritten digits data set, 2013. URL `http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits`.

[9] T. Lyons. Rough paths, Signatures and the modelling of functions on streams. *ArXiv e-prints*, May 2014.

[10] Terry Lyons, Hao Ni, and Harald Oberhauser. A feature set for streams and an application to high-frequency financial tick data. In *Proceedings of the 2014 International Conference on Big Data Science and Computing*, BigDataScience '14, pages 5:1–5:8, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2891-3. doi: 10.1145/2640087.2644157. URL `http://doi.acm.org/10.1145/2640087.2644157`.

[11] T.J. Lyons, M.J. Caruana, and T. Lévy. *Differential Equations Driven by Rough Paths: Ecole dEté de Probabilités de Saint-Flour XXXIV-2004*. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 2007. ISBN 9783540712855. URL `https://books.google.com/books?id=hOm5BQAAQBAJ`.

[12] MATLAB. *R2009a*. The MathWorks Inc., Natick, Massachusetts, 2009. URL `http://www.mathworks.com/help/matlab/`.

[13] MATLAB. *R2016a*. The MathWorks Inc., Natick, Massachusetts, 2016. URL `http://www.mathworks.com/help/matlab/`.