Learning with signatures

Conférence TRAG 2019

Adeline Fermanian

Nancy, October 10th 2019

SORBONNE UNIVERSITÉ

MATH INNOV

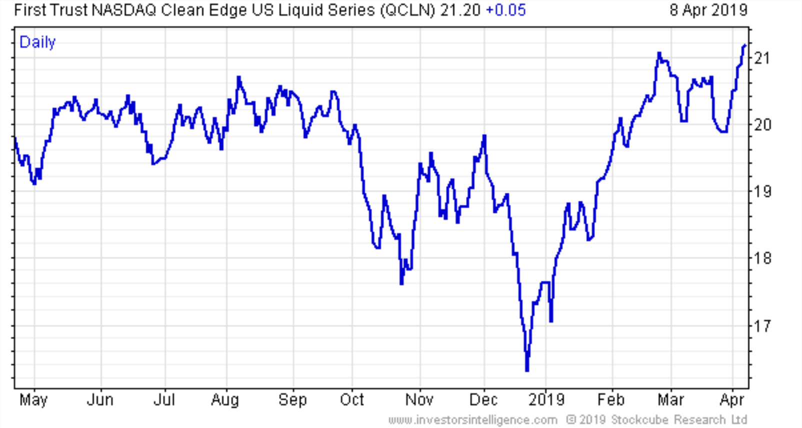
îledeFrance

Supervisors

Benoît Cadre
University Rennes 2

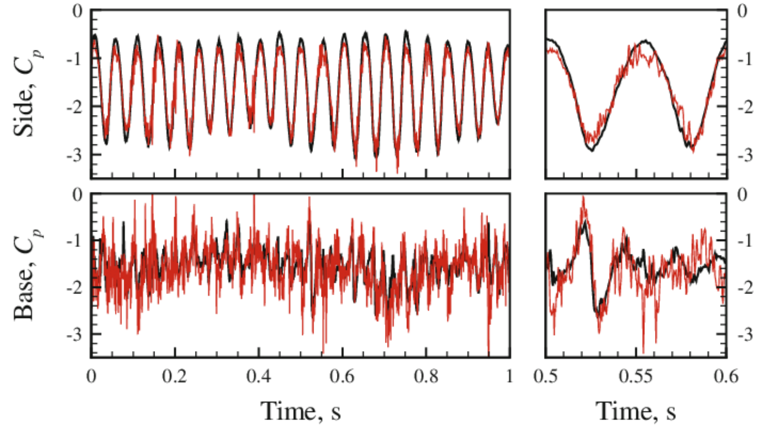
Gérard Biau
Sorbonne University

Learning from a data stream



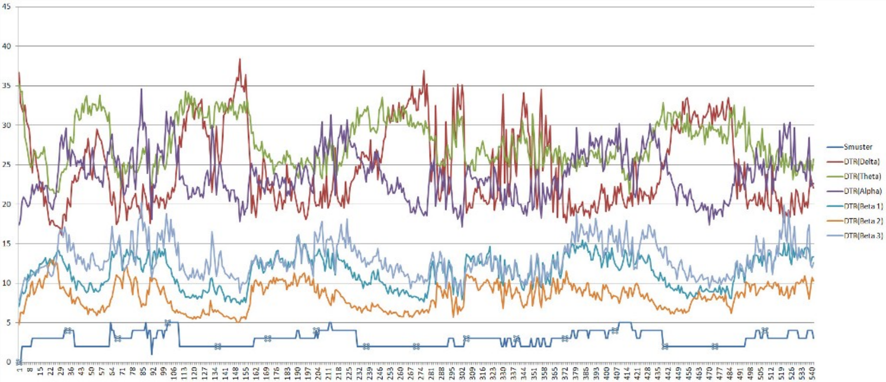
Time series prediction

Learning from a data stream



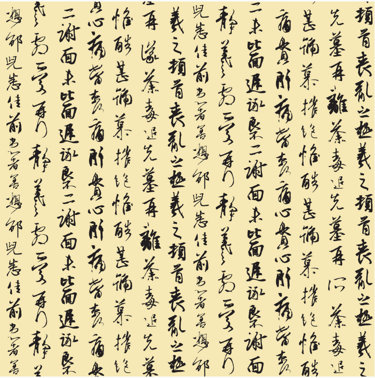
Stereo sound recognition

Learning from a data stream



Automated medical diagnosis from sensor data

Learning from a data stream

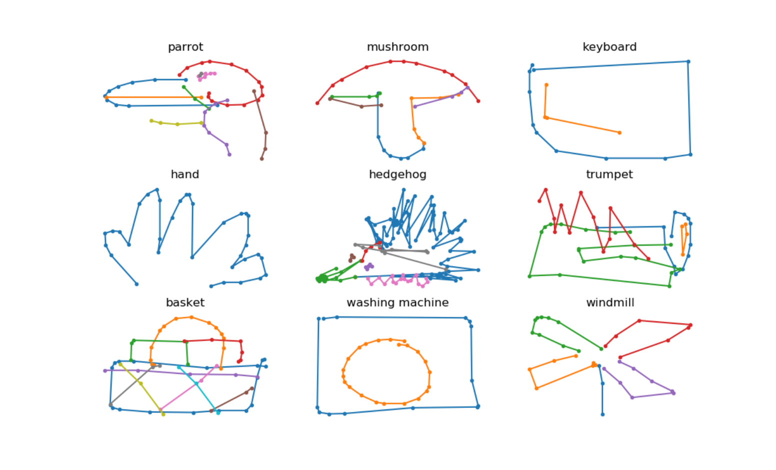


Recognition of characters or handwriting

Common feature

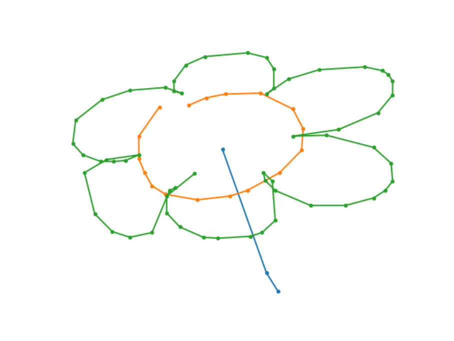
$\boxed{\text{The predictor is a path } X : [a, b] \to \mathbb{R}^d.}$

Google "Quick, Draw!" dataset



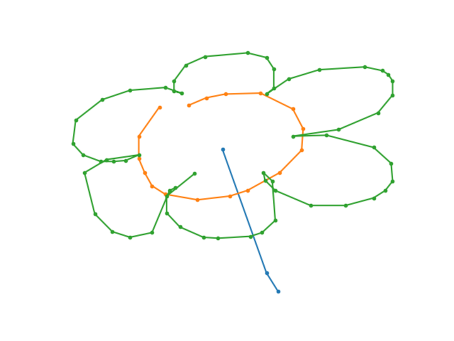
50 million drawings, 340 classes

Data representation

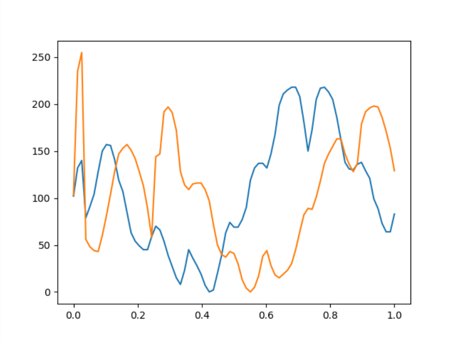


A sample from the class flower

Data representation

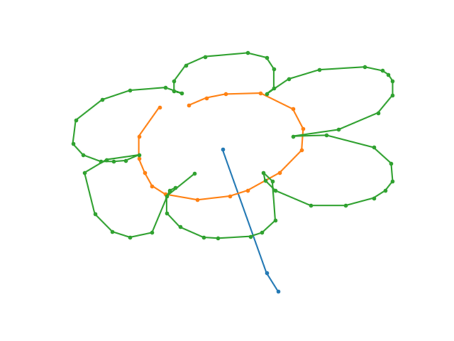


A sample from the class flower

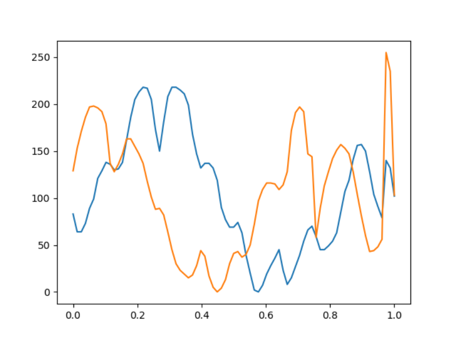


x and y coordinates

Data representation

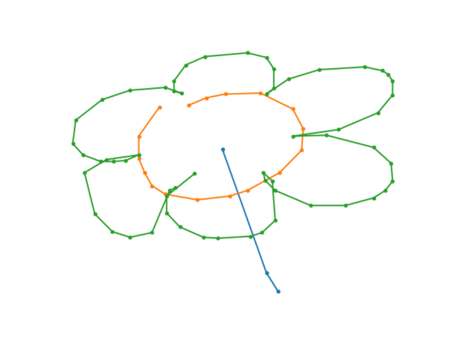


A sample from the class flower

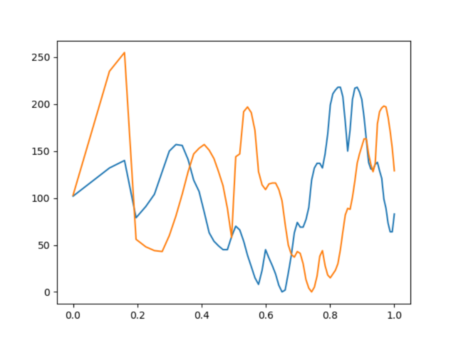


Time reversed

Data representation



A sample from the class flower



x and y at a different speed

The signature will overcome some of these problems.

The signature will overcome some of these problems.

- ▷ It is a transformation from a path to a sequence of coefficients.

The signature will overcome some of these problems.

- It is a transformation from a path to a sequence of coefficients.
- Independent of time parameterization.

The signature will overcome some of these problems.

- It is a transformation from a path to a sequence of coefficients.
- Independent of time parameterization.
- Encodes geometric properties of the path.

The signature will overcome some of these problems.

- It is a transformation from a path to a sequence of coefficients.
- Independent of time parameterization.
- Encodes geometric properties of the path.
- No loss of information.

Table of contents

Definition and basic properties

A brief history

Chen's work for piecewise smooth paths.

ANNALS OF MATHEMATICS
Vol. 65, No. 1, January, 1957
Printed in U.S.A.

INTEGRATION OF PATHS, GEOMETRIC INVARIANTS AND A GENERALIZED BAKER-HAUSDORFF FORMULA

BY KUO-TSAI CHEN

(Received October 17, 1955)

(Revised May 28, 1956)

Let $\alpha: \langle \alpha_1(t), \cdots, \alpha_m(t) \rangle$, $a \leqq t \leqq b$, be a path in the affine m-space R^m. Starting from the line integral $\int_\alpha dx_i$, we define inductively, for $p \geqq 2$,

$$\int_\alpha dx_{i_1} \cdots dx_{i_p} = \int_a^b \left(\int_{\alpha^t} dx_{i_1} \cdots dx_{i_{p-1}} \right) d\alpha_{i_p}(t),$$

where α^t denotes the portion of α with the parameter ranging from a to t. It is observed that $\int_\alpha dx_{i_1} \cdots dx_{i_p}$ acts as a p^{th} order contravariant tensor associated with the path α when R^m undergoes a linear transformation. Some affine and euclidean invariants of α are derived from these tensors. Moreover, we associate to the path α the formal power series

$$\theta(\alpha) = 1 + \textstyle\sum_{p=1}^{\infty} \sum \left(\int_\alpha dx_{i_1} \cdots dx_{i_p} \right) X_{i_1} \cdots X_{i_p}$$

where $X_1, \cdots, X_m$ are noncommutative indeterminates. Theorem 4.2 asserts that $\log \theta(\alpha)$ is a Lie element, i.e., a formal power series $u_1 + \cdots + u_p + \cdots$, where each u_p is a form of degree p generated by $X_1, \cdots, X_m$ through taking bracket products and forming linear combinations. We obtain, as a corollary, the Baker-Hausdorff formula which states that, if X and Y are noncommutative indeterminates, then $\log(\exp X \cdot \exp Y)$ is a Lie element.

Section 1 supplies first some basic knowledge about non-commutative formal power series and then some preparatory definitions and formulas for Theorems 4.1 and 4.2. In Section 2, the iterated integration of paths is defined; and, in Section 3, its geometric applications are indicated. Section 4 contains mainly the proof of the generalized Baker-Hausdorff formula which is further extended, in Section 5, to the case where the affine space R^m is replaced by a differentiable mainfold. For those who are only interested in the geometric aspect of this paper, Sections 2 and 3 may be easily read without Section 1.

This paper is a continuation of the author's work in [Chen, (3)] and is somewhat related to the paper [Chen, (2)]. The proof of Lemma 1.2 is essentially Hausdorff's, in which Lemma 1.1 is implicitly used. Its proof, not an obvious one, is furnished in this paper. Though borrowing some of Hausdorff's technique, Theorem 4.2 is proved in a simpler way and offers a stronger result than the Baker-Hausdorff formula.

A brief history

Lyons' extension to rough paths.

A brief history

Machine learning applications are $\nearrow$.

DeepWriterID: An End-to-end Online Text-independent Writer Identification System

Weixin Yang, Lianwen Jin*, Manfei Liu
College of Electronic and Information Engineering, South China University of Technology, Guangzhou, China
wxy1290@163.com, *lianwen.jin@gmail.com

***Abstract*—Owing to the rapid growth of touchscreen mobile terminals and pen-based interfaces, handwriting-based writer identification systems are attracting increasing attention for personal authentication and digital forensics. However, most studies on writer identification have not been satisfying because of the insufficiency of data and the difficulty of designing good features for various conditions of handwriting samples. Hence, we introduce an end-to-end system called DeepWriterID that employs a deep convolutional neural network (CNN) to address these problems. A key feature of DeepWriterID is a new method we are proposing, called DropSegment. It is designed to achieve data augmentation and to improve the generalized applicability of CNN. For sufficient feature representation, we further introduce path-signature feature maps to improve performance. Experiments were conducted on the NLPR handwriting database. Even though we only use pen-position information in the pen-down state of the given handwriting samples, we achieved new state-of-the-art identification rates of 95.72% for Chinese text and 98.51% for English text.**

***Keywords*—Online text-independent writer identification; convolutional neural network; deep learning; DropSegment; path-signature feature maps.**

1. Introduction

Writer identification is a task of determining a list of candidate writers according to the degree of similarity between their handwriting and a sample of unknown authorship [1]. Currently, it is popular owing to the development and commercialization of touchscreen or pen-enabled electronic devices such as smartphones, and tablet PCs. Its wide range of downstream uses include distinguishing forensic trace evidence, performing mobile bank transactions, and authenticating access to networks. Since most of these applications are closely related to the purpose of assuring personal and property security, handwriting identification merits more attention from academia and industry.

Identifying the handwriting of a writer is one of the highly challenging problems in the fields of artificial intelligence and pattern recognition. Conventionally, handwriting identification systems follow a sequence of data acquisition, data preprocessing, feature extraction, and classification [2]. Research into handwriting identification has been focused on two categories: offline and online. Offline handwritten materials are considered more general but harder to identify, as they contain merely scanned image information. In contrast, systems

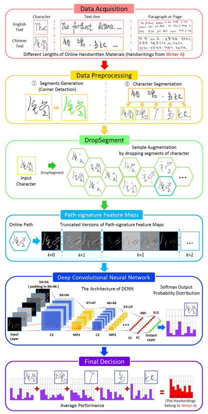


Figure 1. Illustration of DeepWriterID for online handwriting-based writer identification.

Mathematical setting

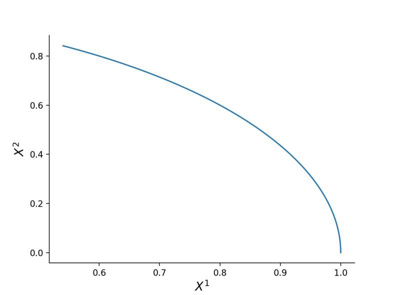
- A path $X : [0,1] \to \mathbb{R}^d$. Notation: X_t.

Mathematical setting

- A path $X : [0,1] \to \mathbb{R}^d$. Notation: X_t.
- Assumption: $\|X\|_{\text{1-var}} < \infty$.

Mathematical setting

- A path $X : [0,1] \to \mathbb{R}^d$. Notation: X_t.
- Assumption: $\|X\|_{\text{1-var}} < \infty$.
- Example: $X_t = (X_t^1, X_t^2) = (\cos t, \sin t),\ t \in [0,1]$.



Path integral

- $X : [0, 1] \to \mathbb{R}$ path of bounded variation.
- $Y : [0, 1] \to \mathbb{R}$ a continuous path.

Path integral

- $X : [0,1] \to \mathbb{R}$ path of bounded variation.
- $Y : [0,1] \to \mathbb{R}$ a continuous path.
- Riemann-Stieljes integral of Y against X

$$\int_0^1 Y_t dX_t$$

is well-defined.

Path integral

- $X : [0,1] \to \mathbb{R}$ path of bounded variation.
- $Y : [0,1] \to \mathbb{R}$ a continuous path.
- Riemann-Stieljes integral of Y against X

$$\int_0^1 Y_t dX_t$$

  is well-defined.
- Example: X_t continuously differentiable:

$$\int_0^1 Y_t dX_t = \int_0^1 Y_t \dot{X}_t dt$$

Path integral

- $X : [0,1] \to \mathbb{R}$ path of bounded variation.
- $Y : [0,1] \to \mathbb{R}$ a continuous path.
- Riemann-Stieljes integral of Y against X

$$\int_0^1 Y_t dX_t$$

  is well-defined.
- Example: $Y_t = 1$ for all $t \in [0,1]$:

$$\int_0^1 Y_t dX_t = \int_0^1 dX_t = X_1 - X_0.$$

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \ldots, X^d)$.

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \ldots, X^d)$.
- For $i \in \{1, \ldots, d\}$,

$$S^i(X)_{[0,t]} = \int_{0<s<t} dX_s^i = X_t^i - X_0^i$$

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \ldots, X^d)$.
- For $i \in \{1, \ldots, d\}$,

$$S^i(X)_{[0,t]} = \int_{0<s<t} dX_s^i = X_t^i - X_0^i \quad \to \text{a path!}$$

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \dots, X^d)$.
- For $i \in \{1, \dots, d\}$,

$$S^i(X)_{[0,t]} = \int_{0<s<t} dX^i_s = X^i_t - X^i_0 \quad \to \text{a path!}$$

- For $(i,j) \in \{1, \dots, d\}^2$,

$$S^{i,j}(X)_{[0,t]} = \int_{0<s<t} S^i(X)_{[0,s]} dX^j_s = \int_{0<r<s<t} dX^i_r dX^j_s$$

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \ldots, X^d)$.
- For $i \in \{1, \ldots, d\}$,

$$S^i(X)_{[0,t]} = \int_{0<s<t} dX^i_s = X^i_t - X^i_0 \quad \to \text{a path!}$$

- For $(i,j) \in \{1, \ldots, d\}^2$,

$$S^{i,j}(X)_{[0,t]} = \int_{0<s<t} S^i(X)_{[0,s]} dX^j_s = \int_{0<r<s<t} dX^i_r dX^j_s \quad \to \text{a path!}$$

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \dots, X^d)$.
- For $i \in \{1, \dots, d\}$,

$$S^i(X)_{[0,t]} = \int_{0<s<t} dX^i_s = X^i_t - X^i_0 \quad \to \text{a path!}$$

- For $(i,j) \in \{1, \dots, d\}^2$,

$$S^{i,j}(X)_{[0,t]} = \int_{0<s<t} S^i(X)_{[0,s]} dX^j_s = \int_{0<r<s<t} dX^i_r dX^j_s \quad \to \text{a path!}$$

- Recursively, for $(i_1, \dots, i_k) \in \{1, \dots, d\}^k$,

$$S^{(i_1,\dots,i_k)}(X)_{[0,t]} = \int_{0<t_1<t_2<\cdots<t_k<t} dX^{i_1}_{t_1} \dots dX^{i_k}_{t_k}.$$

Iterated integrals

- $X : [0,1] \to \mathbb{R}^d$, $X = (X^1, \dots, X^d)$.
- For $i \in \{1, \dots, d\}$,

$$S^i(X)_{[0,t]} = \int_{0<s<t} dX^i_s = X^i_t - X^i_0 \quad \to \text{a path!}$$

- For $(i,j) \in \{1, \dots, d\}^2$,

$$S^{i,j}(X)_{[0,t]} = \int_{0<s<t} S^i(X)_{[0,s]} dX^j_s = \int_{0<r<s<t} dX^i_r dX^j_s \quad \to \text{a path!}$$

- Recursively, for $(i_1, \dots, i_k) \in \{1, \dots, d\}^k$,

$$S^{(i_1,\dots,i_k)}(X)_{[0,t]} = \int_{0<t_1<t_2<\dots<t_k<t} dX^{i_1}_{t_1} \dots dX^{i_k}_{t_k}.$$

- $S^{(i_1,\dots,i_k)}(X)_{[0,1]}$ is the k-fold iterated integral of X along $i_1, \dots, i_k$.

Signature

Definition
The signature of X is the sequence of real numbers

$$S(X) = (1, S^{1}(X), \ldots, S^{d}(X), S^{(1,1)}(X), S^{(1,2)}(X), \ldots).$$

Signature

Definition
The signature of X is the sequence of real numbers

$$S(X) = (1, S^1(X), \ldots, S^d(X), S^{(1,1)}(X), S^{(1,2)}(X), \ldots).$$

- $d = 3 \to (1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, \ldots)$

Signature

Definition
The signature of X is the sequence of real numbers

$$S(X) = (1, S^1(X), \ldots, S^d(X), S^{(1,1)}(X), S^{(1,2)}(X), \ldots).$$

- $d = 3 \to (1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, \ldots)$
- Tensor notation:

$$\mathbf{X^k} = \sum_{(i_1,\ldots,i_k)\subset\{1,\ldots,d\}^k} S^{(i_1,\ldots,i_k)}(X) e_{i_1} \otimes \cdots \otimes e_{i_k}.$$

Signature

Definition
The signature of X is the sequence of real numbers

$$S(X) = (1, S^{1}(X), \ldots, S^{d}(X), S^{(1,1)}(X), S^{(1,2)}(X), \ldots).$$

- $d = 3 \to (1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, \ldots)$
- Tensor notation:

$$\mathbf{X}^{\mathbf{k}} = \sum_{(i_1, \ldots, i_k) \subset \{1, \ldots, d\}^k} S^{(i_1, \ldots, i_k)}(X) e_{i_1} \otimes \cdots \otimes e_{i_k}.$$

- Signature:

$$S(X) = (1, \mathbf{X}^{\mathbf{1}}, \mathbf{X}^{\mathbf{2}}, \ldots, \mathbf{X}^{\mathbf{k}}, \ldots) \in T(\mathbb{R}^d),$$

Signature

Definition
The signature of X is the sequence of real numbers

$$S(X) = (1, S^{1}(X), \dots, S^{d}(X), S^{(1,1)}(X), S^{(1,2)}(X), \dots).$$

- $d = 3 \to (1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33, 111, 112, 113, \dots)$
- Tensor notation:

$$\mathbf{X^k} = \sum_{(i_1,\dots,i_k)\subset\{1,\dots,d\}^k} S^{(i_1,\dots,i_k)}(X) e_{i_1} \otimes \cdots \otimes e_{i_k}.$$

- Signature:

$$S(X) = (1, \mathbf{X^1}, \mathbf{X^2}, \dots, \mathbf{X^k}, \dots) \in T(\mathbb{R}^d),$$

where

$$T(\mathbb{R}^d) = 1 \oplus \mathbb{R}^d \oplus (\mathbb{R}^d)^{\otimes 2} \oplus \cdots \oplus (\mathbb{R}^d)^{\otimes k} \oplus \cdots$$

Example

For $X_t = (X_t^1, X_t^2)$,

$$\mathbf{X}^1 = \begin{pmatrix} \int_0^1 dX_t^1 & \int_0^1 dX_t^2 \end{pmatrix} = \begin{pmatrix} X_1^1 - X_0^1 & X_1^2 - X_0^2 \end{pmatrix}$$

Example

For $X_t = (X_t^1, X_t^2)$,

$$\mathbf{X}^1 = \begin{pmatrix} \int_0^1 dX_t^1 & \int_0^1 dX_t^2 \end{pmatrix} = \begin{pmatrix} X_1^1 - X_0^1 & X_1^2 - X_0^2 \end{pmatrix}$$

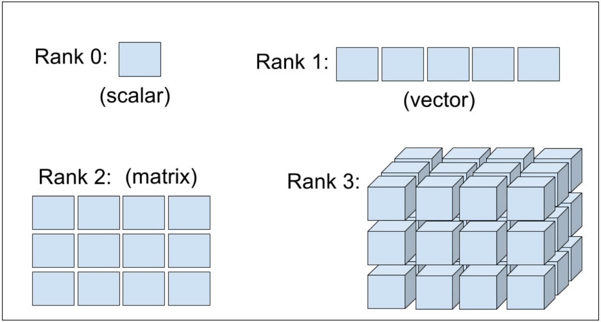
$$\mathbf{X}^2 = \begin{pmatrix} \int_0^1 \int_0^t dX_s^1 dX_t^1 & \int_0^1 \int_0^t dX_s^1 dX_t^2 \\ \int_0^1 \int_0^t dX_s^2 dX_t^1 & \int_0^1 \int_0^t dX_s^2 dX_t^2 \end{pmatrix}$$

Example

For $X_t = (X_t^1, X_t^2)$,

$$\mathbf{X}^1 = \begin{pmatrix} \int_0^1 dX_t^1 & \int_0^1 dX_t^2 \end{pmatrix} = \begin{pmatrix} X_1^1 - X_0^1 & X_1^2 - X_0^2 \end{pmatrix}$$

$$\mathbf{X}^2 = \begin{pmatrix} \int_0^1 \int_0^t dX_s^1 dX_t^1 & \int_0^1 \int_0^t dX_s^1 dX_t^2 \\ \int_0^1 \int_0^t dX_s^2 dX_t^1 & \int_0^1 \int_0^t dX_s^2 dX_t^2 \end{pmatrix}$$



Truncated signature

- Truncated signature at order m:

$$S^m(X) = (1, \mathbf{X^1}, \mathbf{X^2}, \ldots, \mathbf{X^m}).$$

Truncated signature

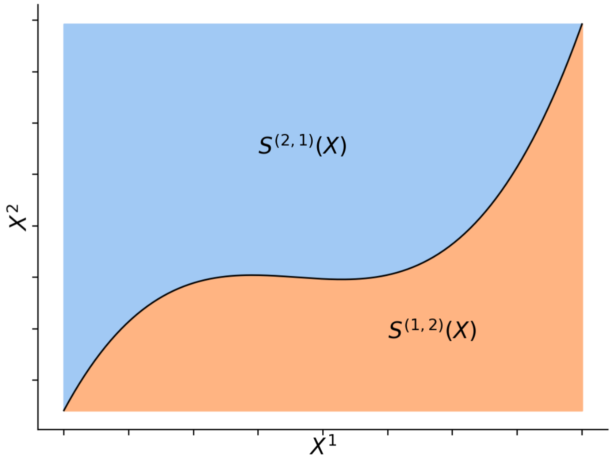
- Truncated signature at order m:

$$S^m(X) = (1, \mathbf{X^1}, \mathbf{X^2}, \dots, \mathbf{X^m}).$$

- Dimension:

$$s_d(m) = \sum_{i=0}^{m} d^i = \frac{d^{m+1} - 1}{d - 1}.$$

Geometric interpretation



Important example

Linear path

- $X : [0, 1] \to \mathbb{R}^d$ a linear path.

Important example

Linear path

- $X : [0,1] \to \mathbb{R}^d$ a linear path.
- $X_t = X_0 + X_1 t$.

Important example

Linear path

- $X : [0,1] \to \mathbb{R}^d$ a linear path.
- $X_t = X_0 + X_1 t$.
- For any $I = (i_1, \ldots, i_k)$,

$$S^I(X) = \frac{1}{k!} \prod_{j=1}^{k} X_1^{i_j}.$$

Important example

Linear path

- $X : [0,1] \to \mathbb{R}^d$ a linear path.
- $X_t = X_0 + X_1 t$.
- For any $I = (i_1, \ldots, i_k)$,

$$S^I(X) = \frac{1}{k!} \prod_{j=1}^{k} X_1^{i_j}.$$

▷ Very useful: in practice, we always deal with piecewise linear paths.

▷ Needed: concatenation operations.

Properties 1

Invariance under time reparametrization

- $X : [0,1] \to \mathbb{R}^d$ a path.

Properties 1

Invariance under time reparametrization

- $X : [0,1] \to \mathbb{R}^d$ a path.
- $\psi : [0,1] \to [0,1]$ a reparametrization

Properties 1

Invariance under time reparametrization

- $X : [0,1] \to \mathbb{R}^d$ a path.
- $\psi : [0,1] \to [0,1]$ a reparametrization
- If $\tilde{X}_t = X_{\psi(t)}$, then

$$S(\tilde{X}) = S(X).$$

Properties 1

Invariance under time reparametrization

- $X : [0,1] \to \mathbb{R}^d$ a path.
- $\psi : [0,1] \to [0,1]$ a reparametrization
- If $\tilde{X}_t = X_{\psi(t)}$, then

$$S(\tilde{X}) = S(X).$$

▷ A key advantage of the signature modeling.

▷ Encoding of the geometric properties of paths.

Properties 2

Chen's identity

- $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$ paths.

Properties 2

Chen's identity

- $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$ paths.
- $X * Y : [a, c] \to \mathbb{R}^d$ the concatenation.

Properties 2

Chen's identity

- $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$ paths.
- $X * Y : [a, c] \to \mathbb{R}^d$ the concatenation.
- Then

$$S(X * Y) = S(X) \otimes S(Y).$$

Properties 2

Chen's identity

- $X : [a, b] \to \mathbb{R}^d$ and $Y : [b, c] \to \mathbb{R}^d$ paths.
- $X * Y : [a, c] \to \mathbb{R}^d$ the concatenation.
- Then
$$S(X * Y) = S(X) \otimes S(Y).$$

▷ We can compute the signature of piecewise linear paths!

▷ Data stream of p points and truncation at m: $O(pd^m)$ operations.

▷ Super fast packages and libraries available in `C++` and `Python`.

Properties 3

Uniqueness results

- `Chen (1958)`: piecewise regular paths.

Properties 3

Uniqueness results

- `Chen (1958)`: piecewise regular paths.
- `Hambly and Lyons (2010)` : paths of bounded variation.

Properties 3

Uniqueness results

- `Chen (1958)`: piecewise regular paths.
- `Hambly and Lyons (2010)` : paths of bounded variation.

  $S(X) = S(Y)$ if and only if the concatenation of X and 'Y run backwards' is a Lipschitz tree-like path.

Properties 3

Uniqueness results

- `Chen (1958)`: piecewise regular paths.
- `Hambly and Lyons (2010)` : paths of bounded variation.

  $S(X) = S(Y)$ if and only if the concatenation of X and 'Y run backwards' is a Lipschitz tree-like path.
- `Boedihardjo et al. (2016)`: extension to weakly geometric rough paths.

Properties 3

Uniqueness results

- Chen (1958): piecewise regular paths.
- Hambly and Lyons (2010) : paths of bounded variation.

  $S(X) = S(Y)$ if and only if the concatenation of X and 'Y run backwards' is a Lipschitz tree-like path.
- Boedihardjo et al. (2016): extension to weakly geometric rough paths.

▷ The signature characterizes paths.

▷ Tools from hyperbolic geometry, Lie groups...

Properties 3

Uniqueness
If X has at least one monotonous coordinate, then $S(X)$ determines X uniquely.

Properties 3

Uniqueness
If X has at least one monotonous coordinate, then $S(X)$ determines X uniquely.

- Trick: add a dummy monotonous component to X.
- Important concept of embedding.

Properties 4

Can we reconstruct the path from its signature ?

- Currently a lot of work in this direction.

Properties 4

Can we reconstruct the path from its signature ?

- Currently a lot of work in this direction.
- A simple procedure has been derived for piecewise linear paths by `Lyons and Xu` (2017)

Properties 4

Can we reconstruct the path from its signature ?

- Currently a lot of work in this direction.
- A simple procedure has been derived for piecewise linear paths by `Lyons and Xu` (2017)

▷ Applications in signal processing, e.g., sound compression.

Properties 5

Signature approximation

- D compact subset of paths from $[0, 1]$ to $\mathbb{R}^d$.

Properties 5

Signature approximation

- D compact subset of paths from $[0, 1]$ to $\mathbb{R}^d$.
- $f : D \to \mathbb{R}$ continuous.

Properties 5

Signature approximation

- D compact subset of paths from $[0,1]$ to $\mathbb{R}^d$.
- $f : D \to \mathbb{R}$ continuous.
- Then, for every $\varepsilon > 0$, there exists $w \in T(\mathbb{R}^d)$ such that, for any $X \in D$,
$$\big| f(X) - \langle w, S(X) \rangle \big| \leq \varepsilon.$$

Properties 5

Signature approximation

- D compact subset of paths from $[0,1]$ to $\mathbb{R}^d$.
- $f : D \to \mathbb{R}$ continuous.
- Then, for every $\varepsilon > 0$, there exists $w \in T(\mathbb{R}^d)$ such that, for any $X \in D$,
$$\big| f(X) - \langle w, S(X) \rangle \big| \leq \varepsilon.$$

▷ Signature and linear model are happy together!

▷ This raises many interesting statistical issues.

Properties 6

Exponential decay of signature coefficients

- $X : [0, 1] \to \mathbb{R}^d$ a path.

Properties 6

Exponential decay of signature coefficients

- $X : [0,1] \to \mathbb{R}^d$ a path.
- Then, for any $k \geq 0$, $I \subset \{1, \dots d\}^k$,

$$|S^I(X)| \leq \frac{\|X\|_{1\text{-var}}^k}{k!}.$$

Properties 6

Exponential decay of signature coefficients

- $X : [0,1] \to \mathbb{R}^d$ a path.
- Then, for any $k \geq 0$, $I \subset \{1, \dots d\}^k$,

$$|S^I(X)| \leq \frac{\|X\|_{1\text{-var}}^k}{k!}.$$

▷ Useful for approximation properties.

Learning with signatures

Parametric supervised machine learning

- Goal: Understand the relationship between an input $X \in \mathcal{X}$ and an output $Y \in \mathcal{Y}$, typically written as

$$Y = f(X) + \varepsilon.$$

Parametric supervised machine learning

- Goal: Understand the relationship between an input $X \in \mathcal{X}$ and an output $Y \in \mathcal{Y}$, typically written as

$$Y = f(X) + \varepsilon.$$

- Data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ i.i.d.

Parametric supervised machine learning

- Goal: Understand the relationship between an input $X \in \mathcal{X}$ and an output $Y \in \mathcal{Y}$, typically written as

$$Y = f(X) + \varepsilon.$$

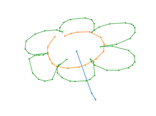
- Data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ i.i.d.
- Prediction function: $f_\theta \approx f$, parameterized by $\theta \in \mathbb{R}^p$.

Parametric supervised machine learning

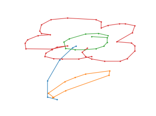
- Goal: Understand the relationship between an input $X \in \mathcal{X}$ and an output $Y \in \mathcal{Y}$, typically written as

$$Y = f(X) + \varepsilon.$$

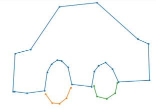
- Data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ i.i.d.
- Prediction function: $f_\theta \approx f$, parameterized by $\theta \in \mathbb{R}^p$.



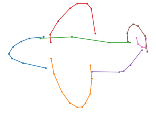
$y_1 = 1$



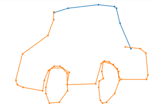
$y_2 = 1$



$y_3 = 2$



$y_4 = 3$



$y_5 = 2$

Parametric supervised machine learning

- Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$.

Parametric supervised machine learning

- Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$.
- Empirical risk minimization: choose

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, f_\theta(x_i)\big).$$

Parametric supervised machine learning

- Loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$.
- Empirical risk minimization: choose

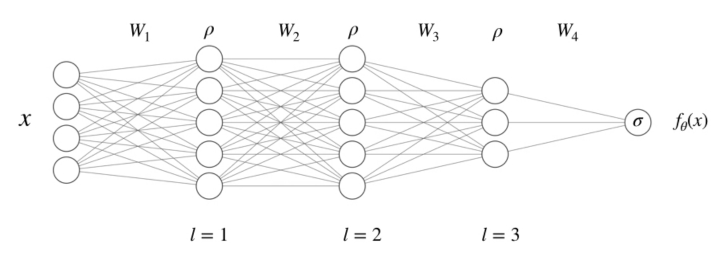
$$\hat{\theta} = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, f_\theta(x_i)\big).$$

Least squares regression

- $\mathcal{X} = \mathbb{R}^p$, $\mathcal{Y} = \mathbb{R}$.
- $f_\theta(x) = \theta^T x$ for any $x \in \mathbb{R}^p$.
- Quadratic loss $\ell(y, f_\theta(x)) = (y - f_\theta(x))^2$

Feedforward neural network

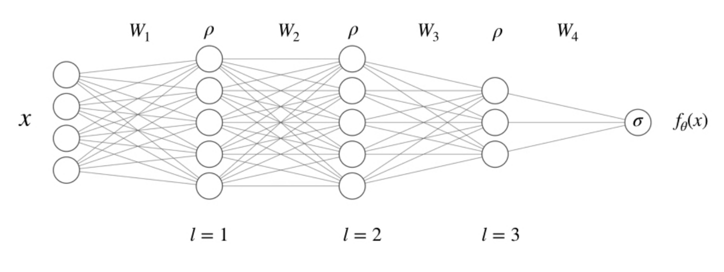
$$f_\theta(x) = \sigma(T_L\rho(T_{L-1}\rho(\ldots\rho(T_1 x))))$$



Feedforward neural network

$$f_\theta(x) = \sigma(T_L\rho(T_{L-1}\rho(\dots\rho(T_1x))))$$

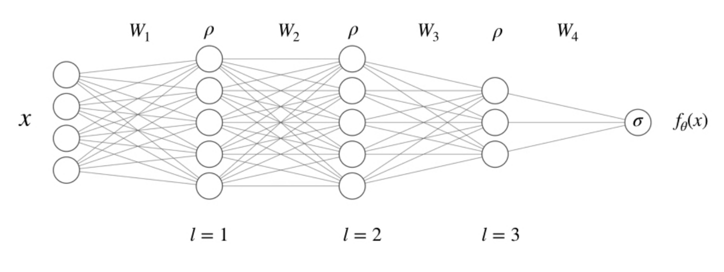
- L layers.



Feedforward neural network

$$f_\theta(x) = \sigma(T_L\rho(T_{L-1}\rho(\ldots\rho(T_1x))))$$

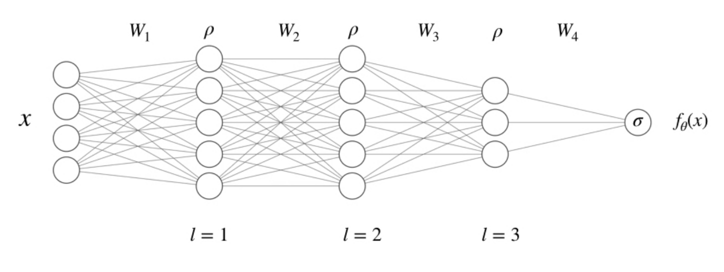
- L layers.
- $\rho : \mathbb{R} \to \mathbb{R}$ activation function (e.g., ReLu $\rho(x) = \max(x, 0)$).



Feedforward neural network

$$f_\theta(x) = \sigma(T_L\rho(T_{L-1}\rho(\ldots\rho(T_1x))))$$

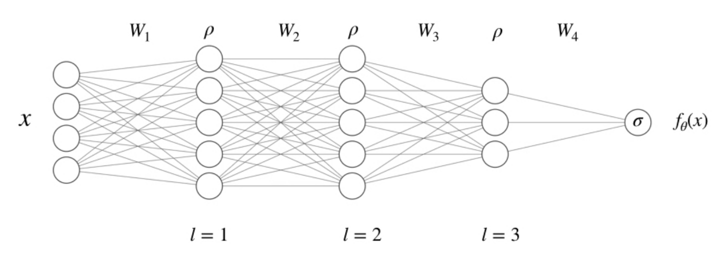
- L layers.
- $\rho : \mathbb{R} \to \mathbb{R}$ activation function (e.g., ReLu $\rho(x) = \max(x, 0)$).
- T_l affine linear transformation between layers:
  $T_lx = W_lx + b_l, \quad l = 1, \ldots, L.$



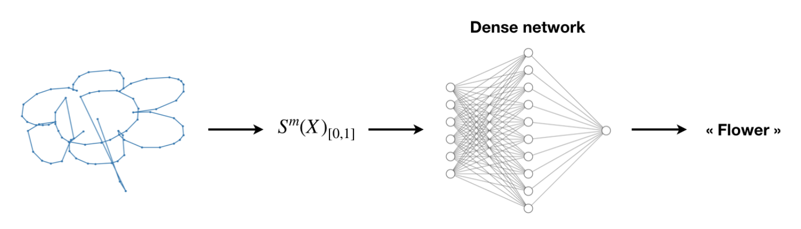
Feedforward neural network

$$f_\theta(x) = \sigma(T_L\rho(T_{L-1}\rho(\dots\rho(T_1x))))$$

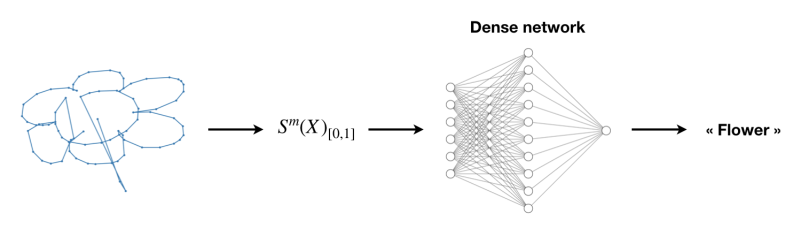
- L layers.
- $\rho : \mathbb{R} \to \mathbb{R}$ activation function (e.g., ReLu $\rho(x) = \max(x, 0)$).
- T_l affine linear transformation between layers:
  $T_l x = W_l x + b_l, \quad l = 1, \dots, L.$
- σ output function.



Signature + learning algorithm

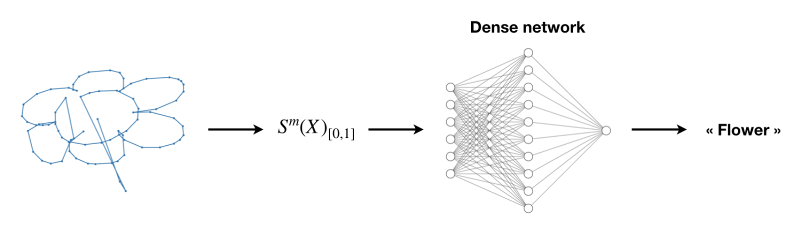


Signature + learning algorithm



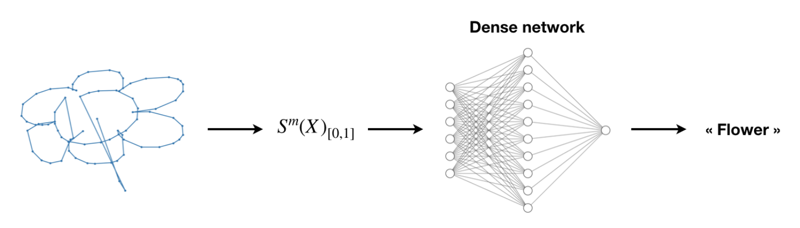
▷ Yang et al. (2017): skeleton-based human action recognition.

Signature + learning algorithm



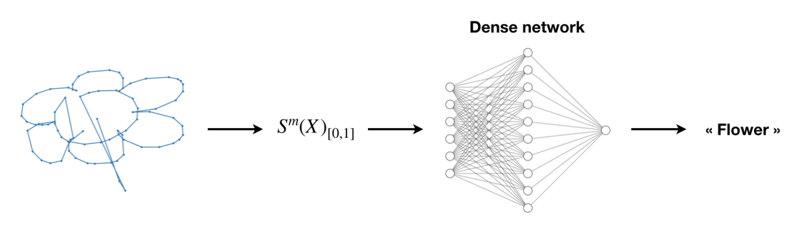
- ▷ Yang et al. (2017): skeleton-based human action recognition.
- ▷ Sequence of positions of human joints

Signature + learning algorithm



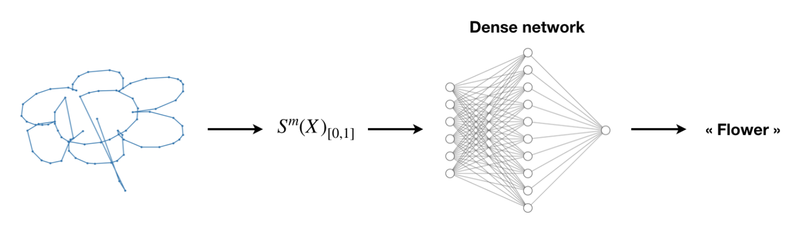
- Yang et al. (2017): skeleton-based human action recognition.
- Sequence of positions of human joints $\to$ high dimensional signature coefficients

Signature + learning algorithm



- ▷ Yang et al. (2017): skeleton-based human action recognition.
- ▷ Sequence of positions of human joints → high dimensional signature coefficients → small dense network

Signature + learning algorithm



- ▷ Yang et al. (2017): skeleton-based human action recognition.
- ▷ Sequence of positions of human joints → high dimensional signature coefficients → small dense network → prediction.

Temporal approaches

- Idea: construct a path of signature coefficients.

Temporal approaches

- Idea: construct a path of signature coefficients.
- Procedure:

Temporal approaches

- Idea: construct a path of signature coefficients.
- Procedure:
  1. Divide the time interval into a partition

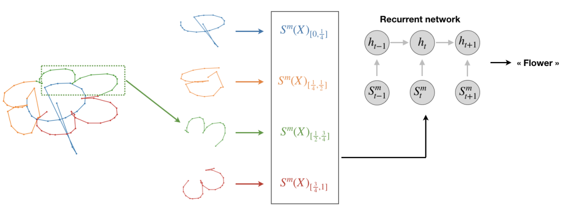
$$0 < 2^{-q} < \cdots < j2^{-q} < \cdots < 1.$$

Temporal approaches

- Idea: construct a path of signature coefficients.
- Procedure:
  1. Divide the time interval into a partition
  $$0 < 2^{-q} < \cdots < j2^{-q} < \cdots < 1.$$
  2. Compute the signature on each interval $[j2^{-q}; (j+1)2^{-q}]$.

Temporal approaches

- Idea: construct a path of signature coefficients.
- Procedure:
  1. Divide the time interval into a partition
$$0 < 2^{-q} < \cdots < j2^{-q} < \cdots < 1.$$
  2. Compute the signature on each interval $[j2^{-q}; (j+1)2^{-q}]$.
  3. The signature sequence S is the input of a recurrent network.



▷ Lai et al. (2017) and Liu et al. (2017): writer recognition.

Recurrent neural network

→ A neural network for sequences.

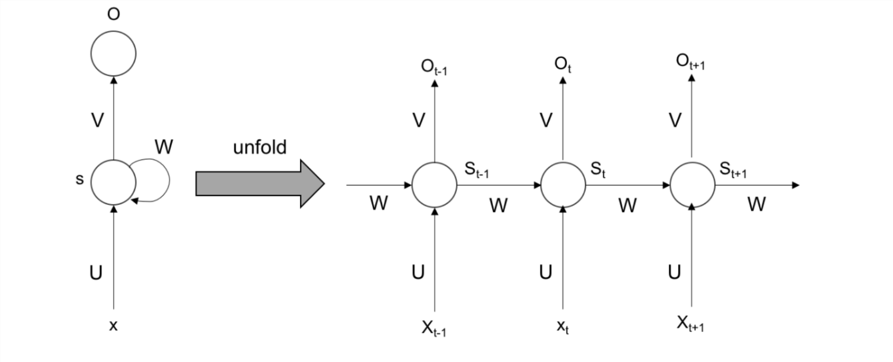


Image approaches

- Idea: create images of signature coefficients.

Image approaches

- Idea: create images of signature coefficients.
- Procedure:

Image approaches

- Idea: create images of signature coefficients.
- Procedure:
    1. Consider that the input path is a trajectory in $\mathbb{R}^2$.

Image approaches

- Idea: create images of signature coefficients.
- Procedure:
  1. Consider that the input path is a trajectory in $\mathbb{R}^2$.
  2. Map this path into an image by forgetting time.

Image approaches

- Idea: create images of signature coefficients.
- Procedure:
  1. Consider that the input path is a trajectory in $\mathbb{R}^2$.
  2. Map this path into an image by forgetting time.
  3. Create one image for each signature coefficient:

Image approaches

- Idea: create images of signature coefficients.
- Procedure:
  1. Consider that the input path is a trajectory in $\mathbb{R}^2$.
  2. Map this path into an image by forgetting time.
  3. Create one image for each signature coefficient:

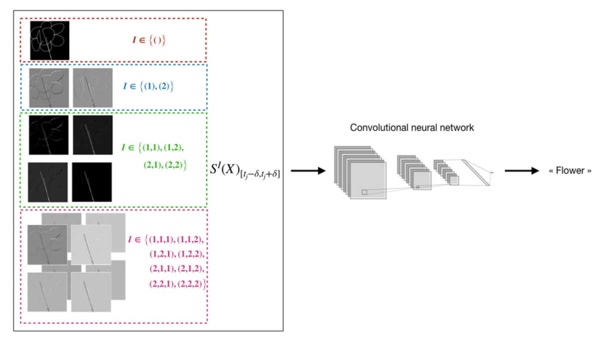
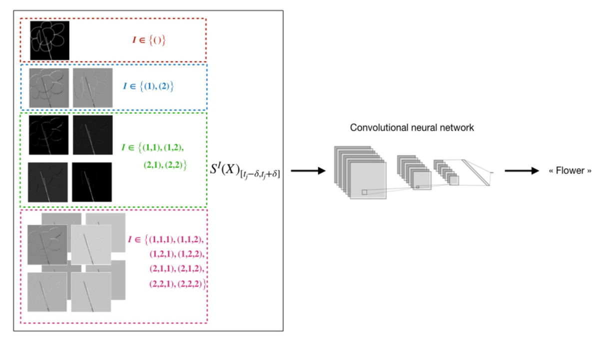


Image approaches

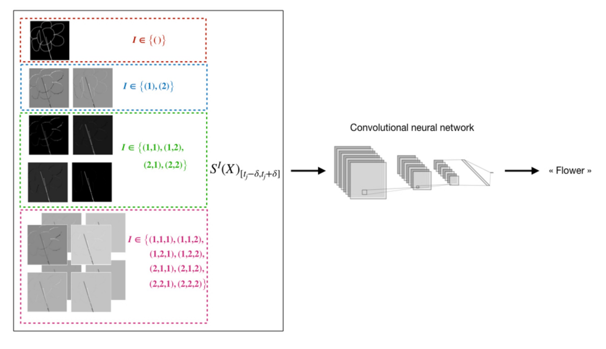
- Idea: create images of signature coefficients.
- Procedure:
    1. Consider that the input path is a trajectory in $\mathbb{R}^2$.
    2. Map this path into an image by forgetting time.
    3. Create one image for each signature coefficient:



    4. This yields $2^{m+1} - 1$ sparse gray pictures.

Image approaches

- Idea: create images of signature coefficients.
- Procedure:
  1. Consider that the input path is a trajectory in $\mathbb{R}^2$.
  2. Map this path into an image by forgetting time.
  3. Create one image for each signature coefficient:

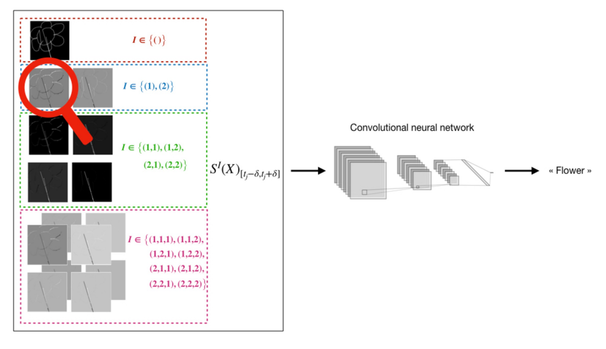


  4. This yields $2^{m+1}-1$ sparse gray pictures.

▷ Graham (2013) and Yang et al. (2016): character recognition.

Image approaches

- Idea: create images of signature coefficients.
- Procedure:
  1. Consider that the input path is a trajectory in $\mathbb{R}^2$.
  2. Map this path into an image by forgetting time.
  3. Create one image for each signature coefficient:



  4. This yields $2^{m+1} - 1$ sparse gray pictures.

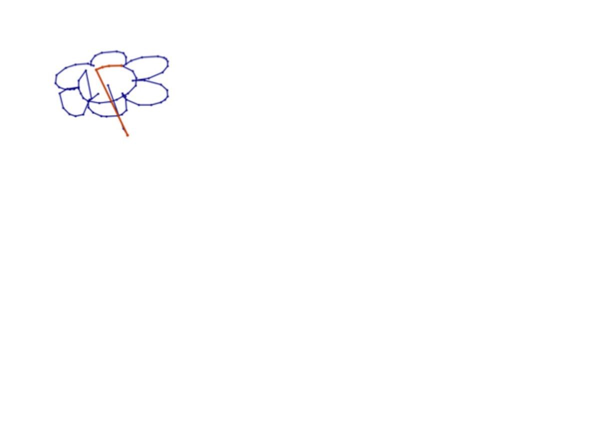
▷ Graham (2013) and Yang et al. (2016): character recognition.

Construction of one signature image

- Procedure:

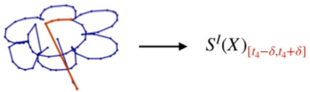
Construction of one signature image

- Procedure:
    1. Consider a window of size $2\delta + 1$ centered at t_i.



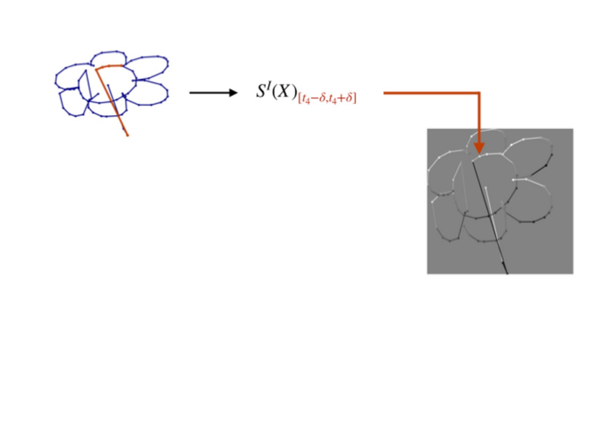
Construction of one signature image

- Procedure:
    1. Consider a window of size $2\delta + 1$ centered at t_i.
    2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.



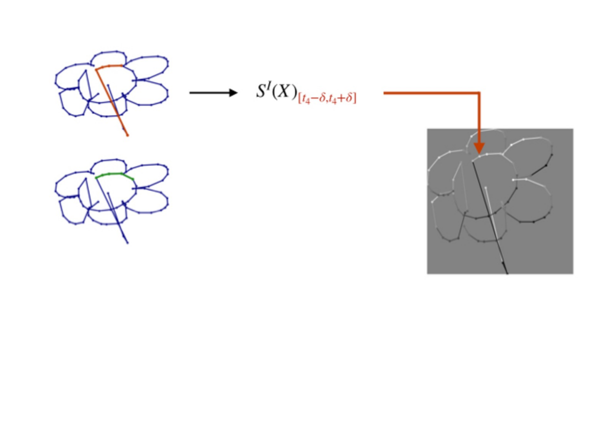
Construction of one signature image

- Procedure:
  1. Consider a window of size $2\delta + 1$ centered at t_i.
  2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.
  3. Store the value obtained as gray level at the pixel corresponding to t_i.



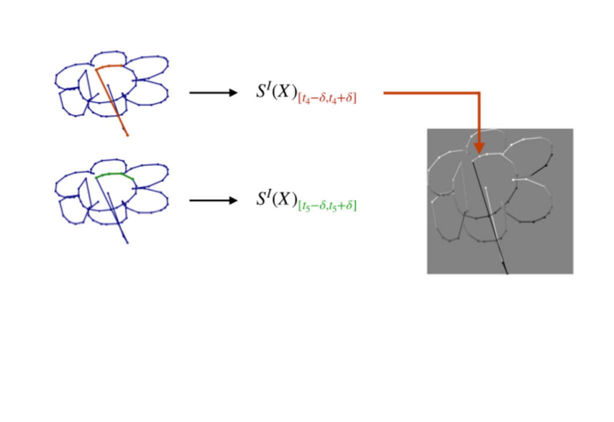
Construction of one signature image

- Procedure:
  1. Consider a window of size $2\delta + 1$ centered at t_i.
  2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.
  3. Store the value obtained as gray level at the pixel corresponding to t_i.
  4. Move the window to the next point and iterate.



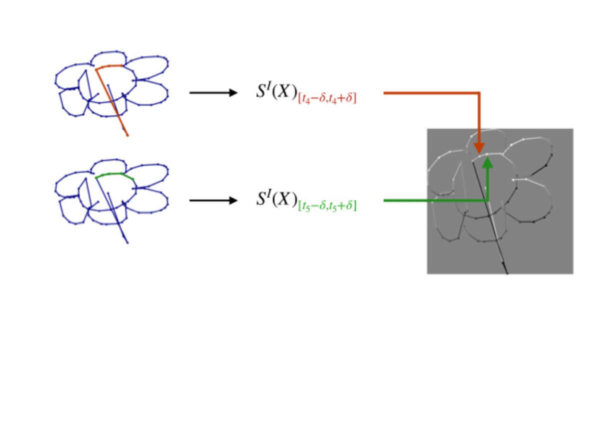
Construction of one signature image

- Procedure:
    1. Consider a window of size $2\delta + 1$ centered at t_i.
    2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.
    3. Store the value obtained as gray level at the pixel corresponding to t_i.
    4. Move the window to the next point and iterate.



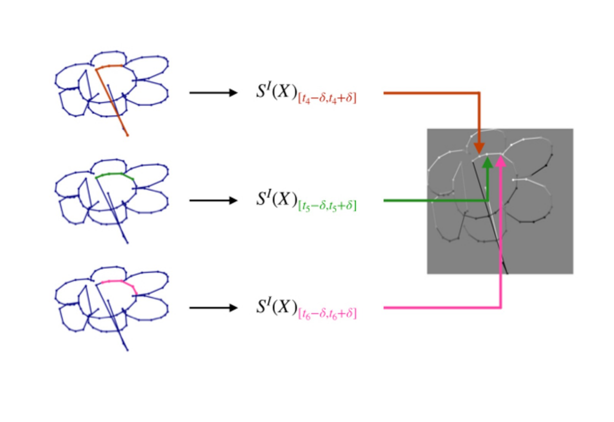
Construction of one signature image

- Procedure:
  1. Consider a window of size $2\delta + 1$ centered at t_i.
  2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.
  3. Store the value obtained as gray level at the pixel corresponding to t_i.
  4. Move the window to the next point and iterate.



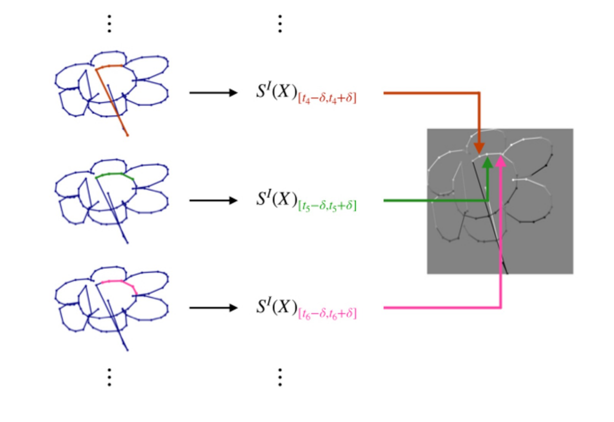
Construction of one signature image

- Procedure:
  1. Consider a window of size $2\delta + 1$ centered at t_i.
  2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.
  3. Store the value obtained as gray level at the pixel corresponding to t_i.
  4. Move the window to the next point and iterate.



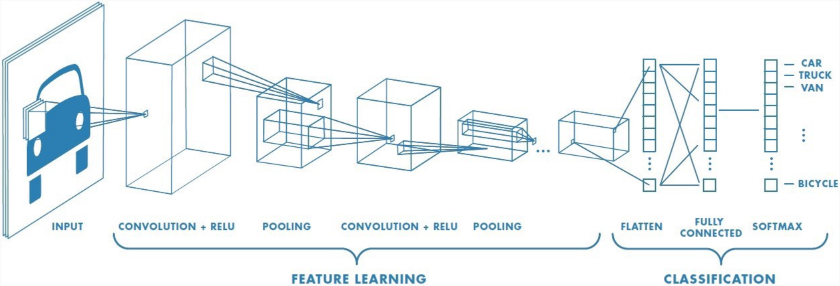
Construction of one signature image

- Procedure:
  1. Consider a window of size $2\delta + 1$ centered at t_i.
  2. Compute the signature coefficient over $[t_i - \delta; t_i + \delta]$.
  3. Store the value obtained as gray level at the pixel corresponding to t_i.
  4. Move the window to the next point and iterate.



Convolutional neural networks

→ A neural network for images.



Questions

$\boxed{\text{Data} \rightarrow \text{Continuous path} \rightarrow \text{Signature} \rightarrow \text{Algorithm.}}$

Questions

$$\boxed{\text{Data} \to \text{Continuous path} \to \text{Signature} \to \text{Algorithm.}}$$

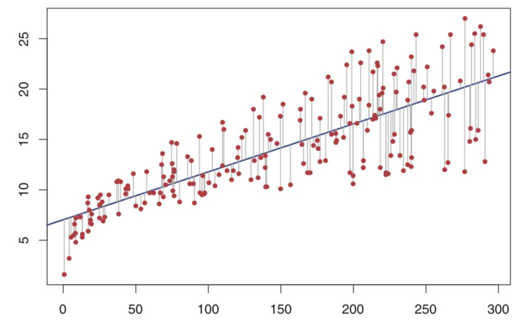
- How should we choose the order of truncation?
- Which path embedding should we use?

Truncation order

Least squares linear regression

Linear model between $x = (x^1, \ldots, x^p) \in \mathbb{R}^p$ and $y \in \mathbb{R}$:

$$y = \beta_0 + \beta_1 x^1 + \cdots + \beta_p x^p + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma^2).$$



Goal: given i.i.d. data $(x_1, y_1), \ldots, (x_n, y_n)$, find $\hat{\beta}$ that minimizes the empirical risk

$$\mathcal{R}_n(\beta) = \sum_{i=1}^{n} (y_i - \beta^T x_i)^2.$$

Regression model on the signature

- $X : [0, 1] \to \mathbb{R}^d$ a path.

Regression model on the signature

- $X : [0,1] \to \mathbb{R}^d$ a path.
- Assumption: there exists $m^* \in \mathbb{N}$, $\beta^* \in \mathbb{R}^{s_d(m^*)}$ such that

$$Y = \langle \beta^*, S^{m^*}(X) \rangle + \varepsilon,$$

where

$$\mathbb{E}(\varepsilon | X) = 0 \quad \text{and} \quad \mathrm{Var}(\varepsilon | X) = \sigma^2 < \infty.$$

Regression model on the signature

- $X : [0,1] \to \mathbb{R}^d$ a path.
- Assumption: there exists $m^* \in \mathbb{N}$, $\beta^* \in \mathbb{R}^{s_d(m^*)}$ such that

$$Y = \langle \beta^*, S^{m^*}(X) \rangle + \varepsilon,$$

  where

$$\mathbb{E}(\varepsilon | X) = 0 \quad \text{and} \quad \mathrm{Var}(\varepsilon | X) = \sigma^2 < \infty.$$

- Goal: estimate β^* and m^*.

Estimation of m^*

- Data: $(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d.

Estimation of m^*

- Data: $(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d.
- For any $k \in \mathbb{N}$, $\alpha > 0$,

$$B_{k,\alpha} = \left\{ \beta \in \mathbb{R}^{s_d(k)} : \|\beta\|_2 \leq \alpha \right\}.$$

Estimation of m^*

- Data: $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d.
- For any $k \in \mathbb{N}$, $\alpha > 0$,

$$B_{k,\alpha} = \left\{ \beta \in \mathbb{R}^{s_d(k)} : \|\beta\|_2 \leq \alpha \right\}.$$

- For any $\beta \in B_{k,\alpha}$,

$$\mathcal{R}_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} \big(Y_i - \langle \beta, S^k(X_i) \rangle \big)^2.$$

Estimation of m^*

- Data: $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d.
- For any $k \in \mathbb{N}$, $\alpha > 0$,

$$B_{k,\alpha} = \left\{ \beta \in \mathbb{R}^{s_d(k)} : \|\beta\|_2 \leq \alpha \right\}.$$

- For any $\beta \in B_{k,\alpha}$,

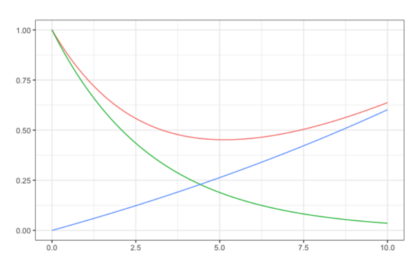
$$\mathcal{R}_n(\beta) = \frac{1}{n} \sum_{i=1}^{n} \big(Y_i - \langle \beta, S^k(X_i) \rangle \big)^2.$$

- For any $k \in \mathbb{N}$,

$$\hat{L}_n(k) = \inf_{\beta \in B_{k,\alpha}} \mathcal{R}_n(\beta).$$

Estimator:

$$\hat{m} = \inf\Big(\operatorname*{argmin}_k \big(\hat{L}_n(k) + \mathrm{pen}_n(k)\big)\Big).$$



Additional assumptions:

(H_0) There exists $K_Y > 0$ such that almost surely $|Y| \leq K_Y$.

(H_1) There exists $K_X > 0$ such that almost surely $\|X\|_{1\text{-var}} \leq K_X$.

Result

Theorem
Let $0 < \rho < \frac{1}{2}$ and

$$\mathsf{pen}_n(k) = K_{\mathsf{pen}} n^{-\rho} \sqrt{d^{k+1} - 1},$$

where $K_{\mathsf{pen}} > 0$ is a constant. Then, under (H_0) and (H_1), for all n large enough,

$$\mathbb{P}(\hat{m} \neq m^*) \leq C_1 \exp(-n^{1-2\rho} C_2).$$

Result

Theorem
Let $0 < \rho < \frac{1}{2}$ and

$$\text{pen}_n(k) = K_{\text{pen}} n^{-\rho} \sqrt{d^{k+1} - 1},$$

where $K_{\text{pen}} > 0$ is a constant. Then, under (H_0) and (H_1), for all n large enough,

$$\mathbb{P}(\hat{m} \neq m^*) \leq C_1 \exp(-n^{1-2\rho} C_2).$$

Corollary
$\hat{m}$ converges almost surely towards m^*.

Path embeddings

Embedding
A way of mapping discrete sequential data into a continuous path.

Kaggle prediction competition

Featured Prediction Competition

Quick, Draw! Doodle Recognition Challenge

How accurately can you identify a doodle?

Google AI · 1,316 teams · 4 months ago

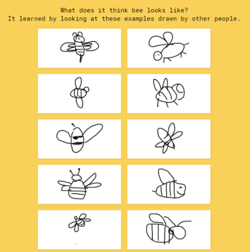
$25,000
Prize Money

Overview | Data | Kernels | Discussion | Leaderboard | Rules | Team | My Submissions | Late Submission

Overview

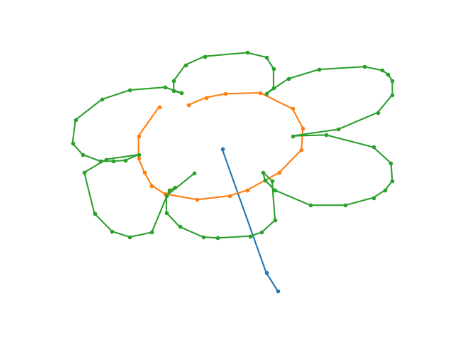
- Description
- Evaluation
- Prizes
- Timeline

"Quick, Draw!" was released as an experimental game to educate the public in a playful way about how AI works. The game prompts users to draw an image depicting a certain category, such as "banana," "table," etc. The game generated more than 1B drawings, of which a subset was publicly released as the basis for this competition's training set. That subset contains 50M drawings encompassing 340 label categories.



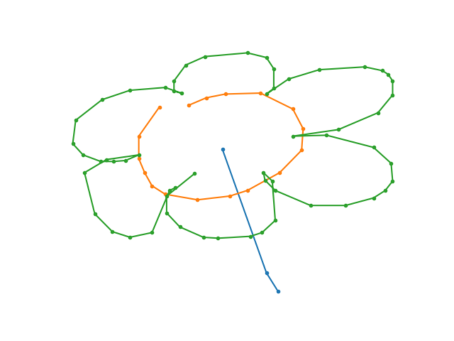
Sounds fun, right? Here's the challenge: since the training data comes from the game itself, drawings can be incomplete or may not match the label. You'll need to build a recognizer that can effectively learn from this noisy data and perform well on a manually-labeled test set from a different distribution.

Different embeddings



Original data

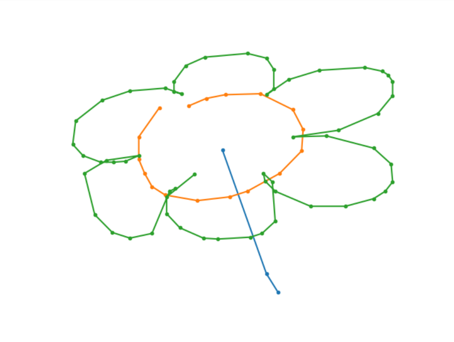
Different embeddings



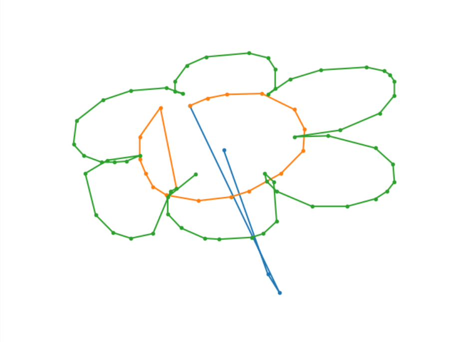
Original data

$$\begin{array}{cc} \text{stroke } 1 & (x_1^1, y_1^1), \ldots, (x_{p_1}^1, y_{p_1}^1) \\ \text{stroke } 2 & (x_1^2, y_1^2), \ldots, (x_{p_2}^2, y_{p_2}^2) \\ \vdots & \vdots \\ \text{stroke } K & (x_1^K, y_1^K), \ldots, (x_{p_K}^K, y_{p_K}^K) \end{array}$$

Different embeddings

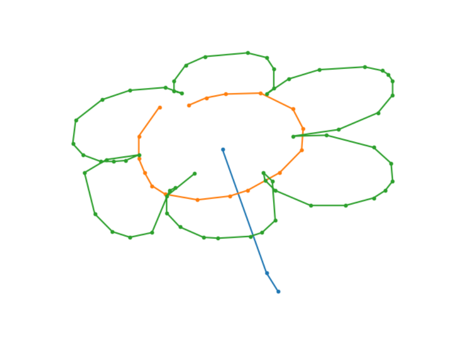


Original data

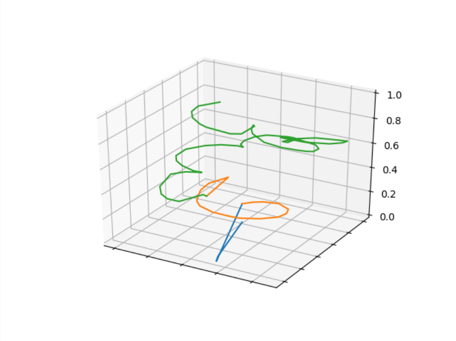


Raw path

Different embeddings

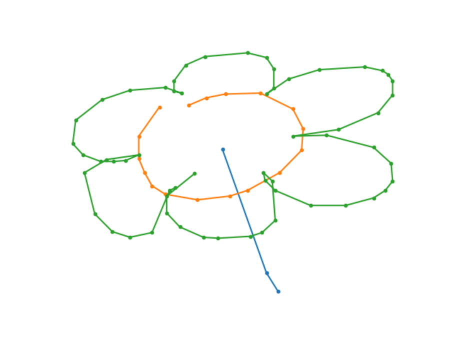


Original data

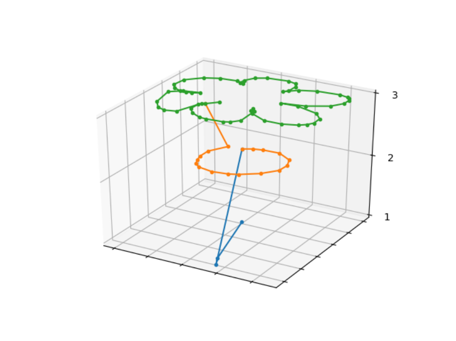


Time path

Different embeddings

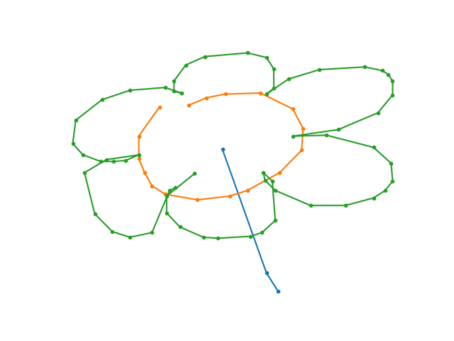


Original data

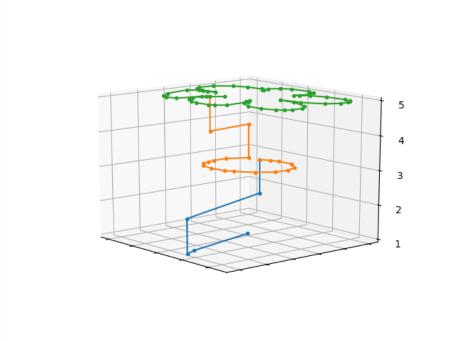


Stroke path

Embedding of path

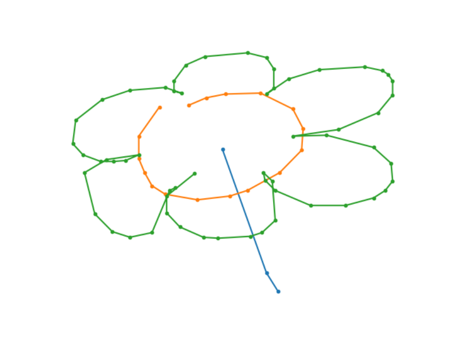


Original data

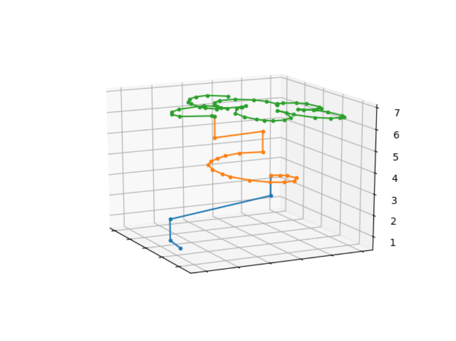


Stroke path, version 2

Embedding of path

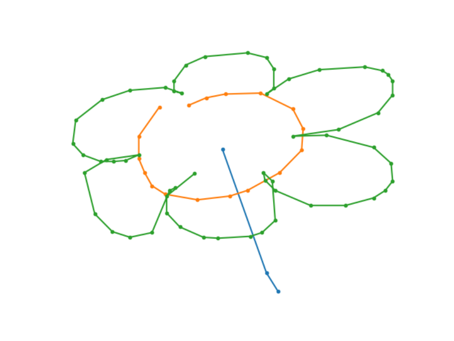


Original data



Stroke path, version 3

Embedding of path



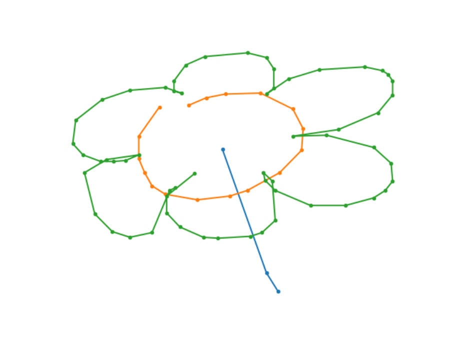
Original data

$t \to (X_t^1, X_t^2, t, X_t^3, X_t^4)$, where

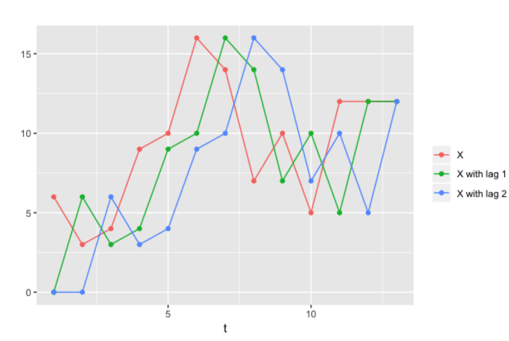
$$X_t^3 = \begin{cases} 0 & \text{if } t < t_1 \\ X_{t-t_1}^1 & \text{otherwise} \end{cases}$$

$$X_t^4 = \begin{cases} 0 & \text{if } t < t_1 \\ X_{t-t_1}^2 & \text{otherwise.} \end{cases}$$

Embedding of path

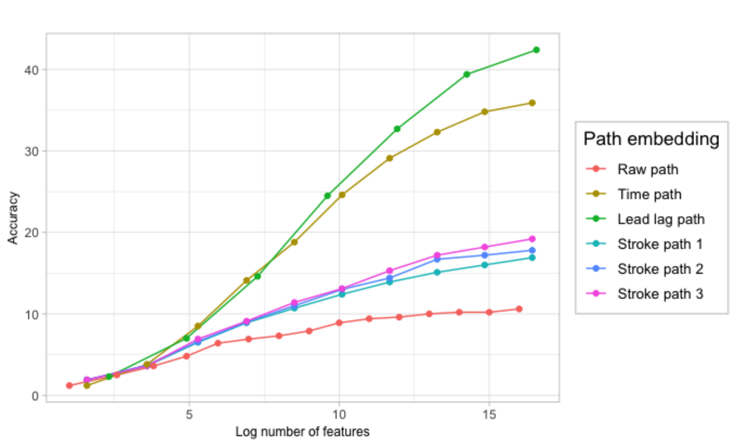


Original data



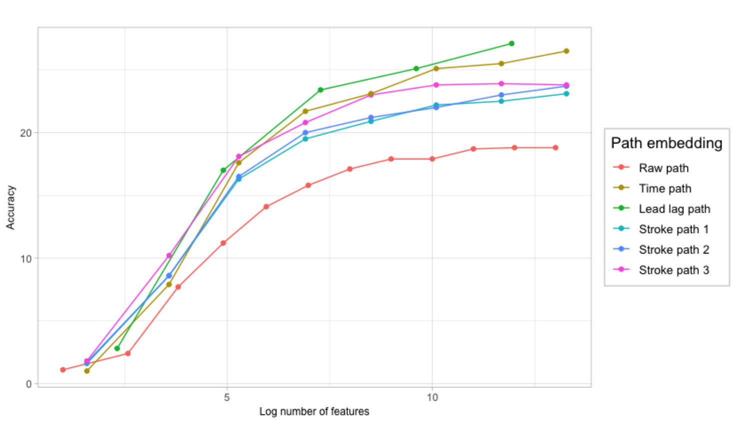
Lead-lag transformation

Quick, Draw! dataset results



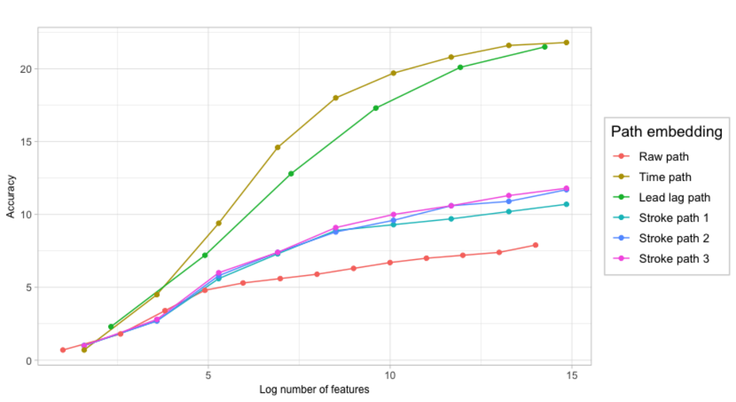
Prediction accuracy with a linear NN

Quick, Draw! dataset results



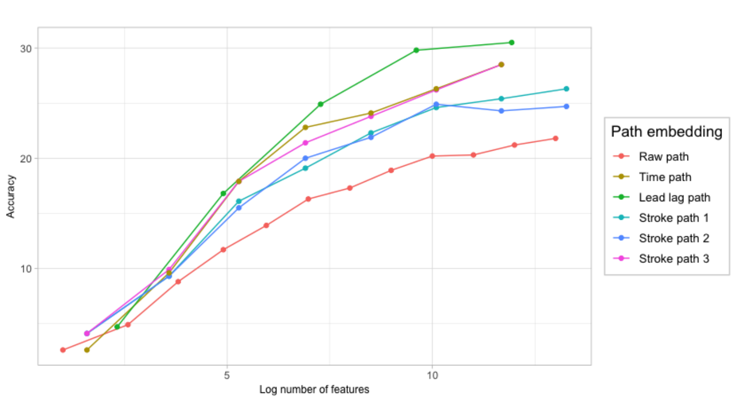
Prediction accuracy with Random Forests

Quick, Draw! dataset results



Prediction accuracy with 5 nearest neighbors

Quick, Draw! dataset results

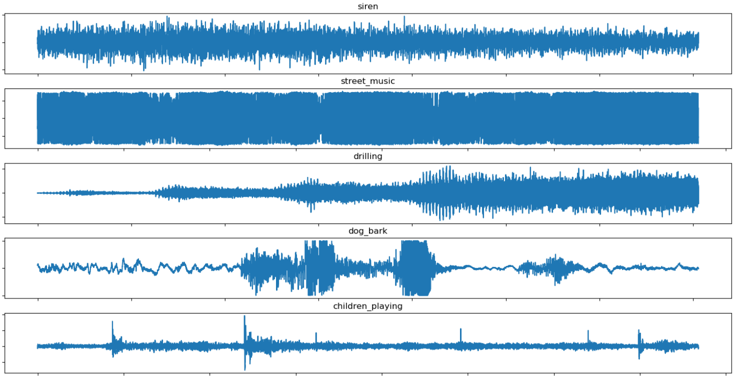


Prediction accuracy with XGBoost

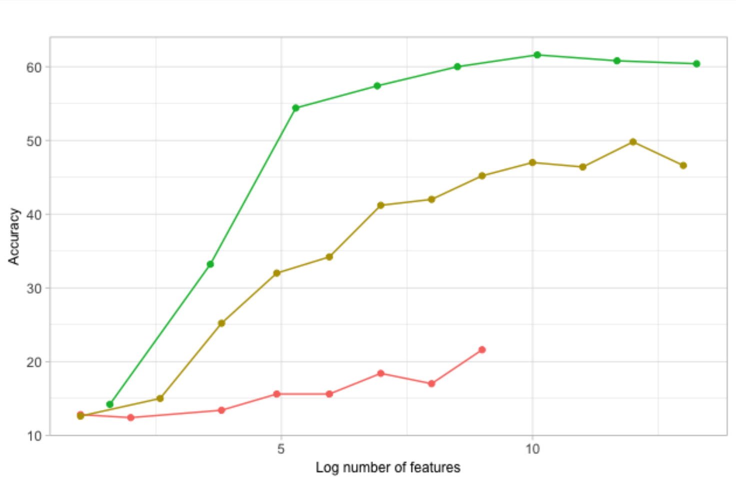
Urban Sound dataset

10 different sounds: car horn, street music, dork barking...

5435 noisy 1-dimensional times series of average size 171 135



Urban Sound dataset results



Prediction accuracy with a linear NN

Motion Sense dataset

Smartphone sensory data recorded by accelerometer and gyroscope sensors

Goal: detect 6 activities (walking upstairs, jogging, sitting...)

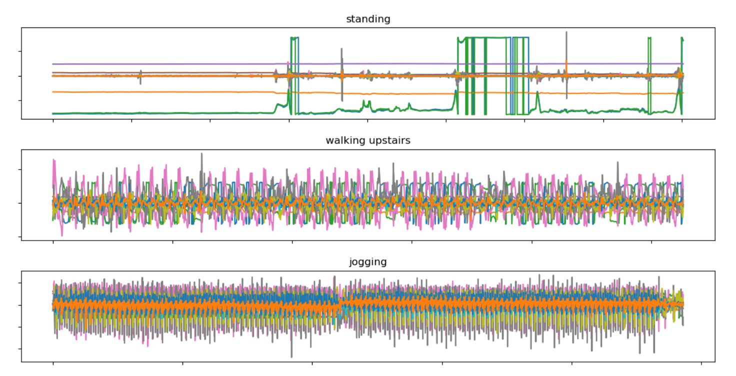
74 800 12-dimensional times series of average size 3934

Motion Sense dataset

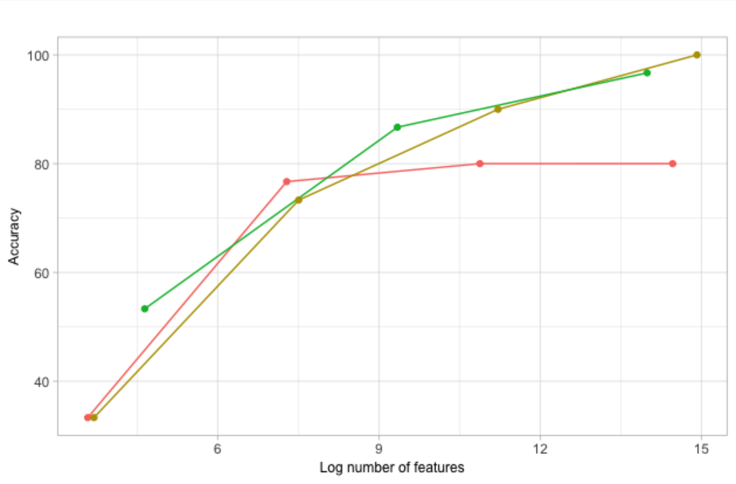
Smartphone sensory data recorded by accelerometer and gyroscope sensors

Goal: detect 6 activities (walking upstairs, jogging, sitting...)

74 800 12-dimensional times series of average size 3934



Motion Sense dataset results



Prediction accuracy with XGBoost

Take-home message

- Striking fact: some embeddings seem consistently better.

Take-home message

- Striking fact: some embeddings seem consistently better.
- Good performance of the lead lag path.

Take-home message

- Striking fact: some embeddings seem consistently better.
- Good performance of the lead lag path.
- This is due to intrinsic properties of the signature and the embedding, not to domain-specific properties.

Take-home message

- Striking fact: some embeddings seem consistently better.
- Good performance of the lead lag path.
- This is due to intrinsic properties of the signature and the embedding, not to domain-specific properties.
- It is particularly remarkable as the dimension of the input stream is different from one dataset to another.

Take-home message

- Striking fact: some embeddings seem consistently better.
- Good performance of the lead lag path.
- This is due to intrinsic properties of the signature and the embedding, not to domain-specific properties.
- It is particularly remarkable as the dimension of the input stream is different from one dataset to another.
- Conclusion: the lead lag embedding seems to be the best choice, regardless of the data and algorithm used.

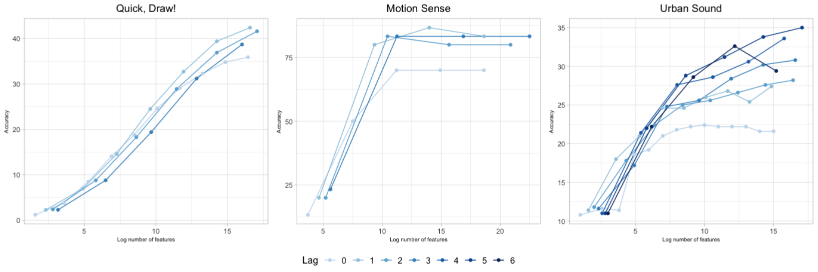
Take-home message

- Striking fact: some embeddings seem consistently better.
- Good performance of the lead lag path.
- This is due to intrinsic properties of the signature and the embedding, not to domain-specific properties.
- It is particularly remarkable as the dimension of the input stream is different from one dataset to another.
- Conclusion: the lead lag embedding seems to be the best choice, regardless of the data and algorithm used.
- Computationally cheap and drastically improves prediction accuracy.

Performance of signatures

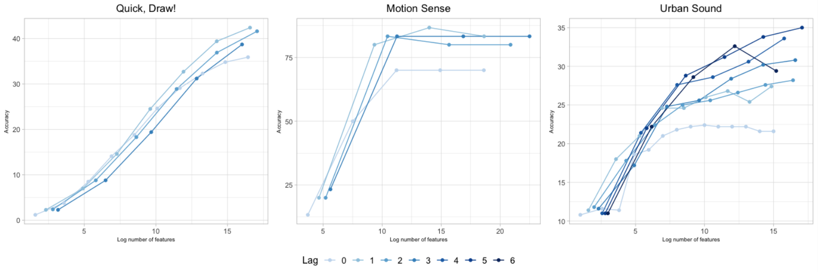
Our plan

- For each dataset: lead lag + lag selection.



Our plan

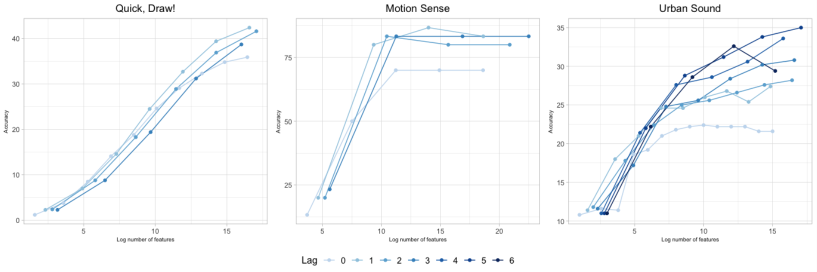
- For each dataset: lead lag + lag selection.



- Quick, Draw! and Motion Sense: 1.

Our plan

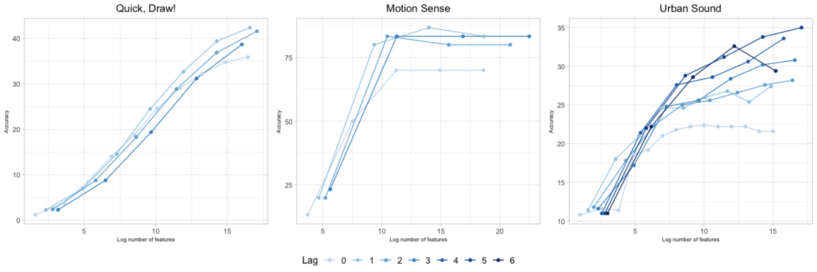
- For each dataset: lead lag + lag selection.



- Quick, Draw! and Motion Sense: 1. Urban Sound: 5.

Our plan

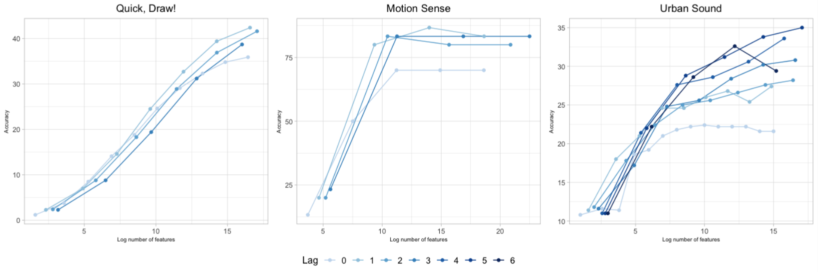
- For each dataset: lead lag + lag selection.



- Quick, Draw! and Motion Sense: 1. Urban Sound: 5.
- Quick, Draw!: dense NN with two hidden layers and ReLu activation (1 523 200 samples for training and 76 160 for validation and test).

Our plan

- For each dataset: lead lag + lag selection.



- Quick, Draw! and Motion Sense: 1. Urban Sound: 5.
- Quick, Draw!: dense NN with two hidden layers and ReLu activation (1 523 200 samples for training and 76 160 for validation and test).
- Urban Sound and Motion Sense: Random Forests.

Performance of signature learning

- Quick, Draw!

Performance of signature learning

- Quick, Draw!
  - State of the art: deep CNN trained with several million of samples.

Performance of signature learning

- Quick, Draw!
  - ▷ State of the art: deep CNN trained with several million of samples.
  - ▷ Kaggle top 3 accuracy = 95%.

Performance of signature learning

- Quick, Draw!
    - State of the art: deep CNN trained with several million of samples.
    - Kaggle top 3 accuracy = 95%.
    - Our result: small NN + signature features at order 6 = 73%.

Performance of signature learning

- Quick, Draw!
  - ▷ State of the art: deep CNN trained with several million of samples.
  - ▷ Kaggle top 3 accuracy $= 95\%$.
  - ▷ Our result: small NN + signature features at order 6 $= 73\%$.
- Urban Sound

Performance of signature learning

- Quick, Draw!
  - State of the art: deep CNN trained with several million of samples.
  - Kaggle top 3 accuracy = 95%.
  - Our result: small NN + signature features at order 6 = 73%.
- Urban Sound
  - State of the art: feature extraction with mixture of experts models.

Performance of signature learning

- Quick, Draw!
  - ▷ State of the art: deep CNN trained with several million of samples.
  - ▷ Kaggle top 3 accuracy $= 95\%$.
  - ▷ Our result: small NN $+$ signature features at order $6 = 73\%$.
- Urban Sound
  - ▷ State of the art: feature extraction with mixture of experts models.
  - ▷ Accuracy $= 77.36\%$.

Performance of signature learning

- Quick, Draw!
  - State of the art: deep CNN trained with several million of samples.
  - Kaggle top 3 accuracy = 95%.
  - Our result: small NN + signature features at order 6 = 73%.
- Urban Sound
  - State of the art: feature extraction with mixture of experts models.
  - Accuracy = 77.36%.
  - Our result: Random Forests + signature features at order 5 = 66.8%.

Performance of signature learning

- Quick, Draw!
  - State of the art: deep CNN trained with several million of samples.
  - Kaggle top 3 accuracy $= 95\%$.
  - Our result: small NN $+$ signature features at order $6 = 73\%$.
- Urban Sound
  - State of the art: feature extraction with mixture of experts models.
  - Accuracy $= 77.36\%$.
  - Our result: Random Forests $+$ signature features at order $5 = 66.8\%$.
- Motion Sense

Performance of signature learning

- Quick, Draw!
  - ▷ State of the art: deep CNN trained with several million of samples.
  - ▷ Kaggle top 3 accuracy = 95%.
  - ▷ Our result: small NN + signature features at order 6 = 73%.
- Urban Sound
  - ▷ State of the art: feature extraction with mixture of experts models.
  - ▷ Accuracy = 77.36%.
  - ▷ Our result: Random Forests + signature features at order 5 = 66.8%.
- Motion Sense
  - ▷ State of the art: deep NN + autoencoders + multi-objective loss.

Performance of signature learning

- Quick, Draw!
    - State of the art: deep CNN trained with several million of samples.
    - Kaggle top 3 accuracy = 95%.
    - Our result: small NN + signature features at order 6 = 73%.
- Urban Sound
    - State of the art: feature extraction with mixture of experts models.
    - Accuracy = 77.36%.
    - Our result: Random Forests + signature features at order 5 = 66.8%.
- Motion Sense
    - State of the art: deep NN + autoencoders + multi-objective loss.
    - F1 score = 92.91.

Performance of signature learning

- Quick, Draw!
  - State of the art: deep CNN trained with several million of samples.
  - Kaggle top 3 accuracy $= 95\%$.
  - Our result: small NN $+$ signature features at order 6 $= 73\%$.
- Urban Sound
  - State of the art: feature extraction with mixture of experts models.
  - Accuracy $= 77.36\%$.
  - Our result: Random Forests $+$ signature features at order 5 $= 66.8\%$.
- Motion Sense
  - State of the art: deep NN $+$ autoencoders $+$ multi-objective loss.
  - F1 score $= 92.91$.
  - Our result: Random Forests $+$ signature features at order 3: 91.

Conclusion

- Our algorithms have not been tuned to a specific application.

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.
- They require less computing resources and no domain-specific knowledge.

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.
- They require less computing resources and no domain-specific knowledge.
- A lot of open questions

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.
- They require less computing resources and no domain-specific knowledge.
- A lot of open questions
  - Theoretical results on the lead lag path.

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.
- They require less computing resources and no domain-specific knowledge.
- A lot of open questions
  - Theoretical results on the lead lag path.
  - Sparsity in signatures.

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.
- They require less computing resources and no domain-specific knowledge.
- A lot of open questions
    - ▷ Theoretical results on the lead lag path.
    - ▷ Sparsity in signatures.
    - ▷ Extension to paths of finite p-variation for $p \geq 2$

Conclusion

- Our algorithms have not been tuned to a specific application.
- However, they achieve results close to state-of-the-art.
- They require less computing resources and no domain-specific knowledge.
- A lot of open questions
  - ▷ Theoretical results on the lead lag path.
  - ▷ Sparsity in signatures.
  - ▷ Extension to paths of finite p-variation for $p \geq 2$
  - ▷ ...

Thank you!

Images taken from

- "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- "Hands-On Natural Language Processing with Python" by Rajalingappaa Shanmugamani, Rajesh Arumugam
- https://towardsdatascience.com "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way"