Regularity structures and machine learning

Ilya Chevyrev

(Joint work with Andris Gerasimovičs & Hendrik Weber)

arXiv:2108.05879

The University of Edinburgh

13 January 2022

DataSig Seminar

Overview

1. Background - signatures
2. Higher dimensions - regularity structures
3. Numerical experiments

Background - signatures

Machine learning

(Simplistic) picture of machine learning:

data $\to$ features $\to$ learning algorithm $\to$ output

- data $\to$ features: vectorisation, dimensional reduction, etc.
- features $\to$ learning algorithm: ‘black box’
- learning algorithm $\to$ output: e.g. response vector, classification label, etc.

Focus: data $\to$ features for data defined on *spatial domains* $D \subset \mathbb{R}^d$

$$\xi \colon D \to \mathbb{R}^n \, .$$

Motivating problem (supervised learning)

From observed samples, ‘learn’ solution to $\mathcal{L}u = \mu(u) + \sigma(u)\xi$.

Naive approach

Discretize D to $\{x_i\}_{i=1}^{N} \subset D$ and use $\{\xi(x_i)\}_{i=1}^{N}$ as a feature vector.

Problems:

- Often needs N very large to be descriptive.
  - Huge computational cost.
- Can be unstable to noise.
- Don't have access to $\{\xi(x)\}_{x \in D}$, only some 'observed points'.
  - discretisations need to know about 'observed points',
  - 'observed points' may vary sample to sample $\Rightarrow$ feature vectors $\{\xi(x_i)\}_{i=1}^{N}$ have different dimensions and not directly comparable.

One-dimensional case - signature

Definition

Consider a (piecewise smooth) $X = (X^1 \dots, X^n) \colon [0, T] \to \mathbb{R}^n$. The *signature* of X is the family of numbers

$$(S(X)^{i_1,\dots,i_k})_{k\geq 0,\ 1\leq i_1,\dots,i_k\leq n}$$

where

$$S(X)^{i_1,\dots i_k} = \int_0^T \int_0^{t_k} \dots \int_0^{t_2} \mathrm{d}X^{i_1}_{t_1} \dots \mathrm{d}X^{i_{k-1}}_{t_{k-1}} \, \mathrm{d}X^{i_k}_{t_k} \, .$$

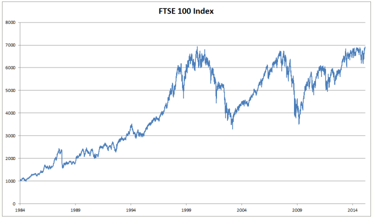
Chen, Ree, Magnus 50's, Brockett, Sussmann, Fliess 70's+, Lyons '90's+

Properties:

- expansions of ODEs $\mathrm{d}Y = \sigma(Y)\,\mathrm{d}X$,
- geometric description of X,
- algebraic properties: generalises polynomials (shuffle product) $\Rightarrow$ 'universal' feature set,
- stable under natural metrics (rough paths).

The signature transform helps analyse **time-ordered data**:

- Financial times-series.



- Text: "*The quick brown fox jumped over the lazy dog.*"

- Time-evolving network.

Grandjean 2014 *Les Cahiers du Numérique*

Example applications

Signatures have been

- combined with convolutional neural nets to win first prize in ICDAR 2013 Online Isolated Chinese Character recognition competition.[1]
- combined with gradient boosting regression to win first prize in PhysioNet 2019 Computing in Cardiology Challenge.[2]
- implemented in Python libraries (on GitHub): iisignature (Graham–Reizenstein), Signatory & ESig (Kidger, Lyons et al.).
- applied to gesture recognition, financial data analysis, neural networks, topological data analysis, hypothesis testing, ...

For a 'primer', see C.–Kormilitzin '16.[3]

---

[1] Benjamin Graham. "Sparse arrays of signatures for online character recognition". *arXiv e-prints*, arXiv:1308.0371 (2013).

[2] J. Morrill et al. "The Signature-Based Model for Early Detection of Sepsis From Electronic Health Records in the Intensive Care Unit". *2019 Computing in Cardiology (CinC)*. 2019.

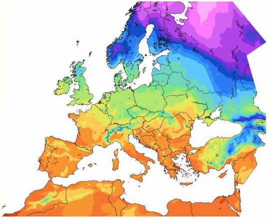
[3] Ilya Chevyrev and Andrey Kormilitzin. "A Primer on the Signature Method in Machine Learning". *arXiv e-prints*, arXiv:1603.03788 (2016).

However, signatures not directly applicable to **spatial data**:

- Image recognition.

RSSCN7 dataset [Zou et al. 2015]

- Meteorological data.



ECMWF 2011

Higher dimensions - regularity structures

How to generalise signatures to higher dimensions?

- Rough paths generalise to regularity structures.
- Basic objects in regularity structures are *models*.

Definition (Model)

Consider $D \subset \mathbb{R}^d$ and linear operator I mapping space of functions $\{u \colon D \to \mathbb{R}\}$ to itself.

Consider further an input $(\{u^i\}_{i=1}^{\ell}, \xi)$ of functions $\xi, u^i \colon D \to \mathbb{R}$.
The *model feature vector* is the family of functions $\cup_{n \geq 0} \mathcal{M}^n$

$$\mathcal{M}^0 = \{u^i\}_{i=1}^{\ell} \qquad \text{(initialising set)},$$

$$\mathcal{M}^n = \Big\{ I\big[\xi^j \prod_{i=1}^{k} \partial^{a_i} f_i\big] \;:\; f_i \in \mathcal{M}^{n-1}, a_i \in \mathbb{N}^d, j, k \in \mathbb{N} \Big\} \cup \mathcal{M}^{n-1}.$$

Think: each $f \in \mathcal{M}$ is indexed by corresponding symbol (tree).

Motivation

Example (Signature)

- Let $X\colon [0,T] \to \mathbb{R}$ and $\xi := \dot{X}$. Define $I[\xi]_t = \int_0^t \xi_s \, \mathrm{d}s$.
- Starting with $\mathcal{M}^0 = \emptyset$, functions in $\mathcal{M}$ evaluated at T encode the signature of X.
  - (Works also for $X\colon [0,T] \to \mathbb{R}^n$.)

Example (PDEs)

Suppose we want to approximate the solution $u\colon D \to \mathbb{R}$ to

$$\mathcal{L}u = \mu(u, \nabla u) + \sigma(u, \nabla u)\xi \,, \qquad u\big|_{\partial D} = u_0 \,,$$

- $\mathcal{L}$ is a differential operator, μ, σ are (smooth/analytic) functions,
- (ξ, u_0) is the **input**.
  - (More boundary conditions could be necessary.)

Example (PDEs cont.)

Picard's theorem: $u = \lim_{n\to\infty} u^n$ where $u^0 = I_c[u_0]$ and

$$u^{n+1} = I_c[u_0] + I[\mu(u^n)] + I[\sigma(u^n)\xi] \,, \quad \text{and}$$

$$\Big\{\mathcal{L}I[f] = f \,, \quad I[f]\big|_{\partial D} = 0 \,, \quad \Big\{\mathcal{L}I_c[g] = 0 \,, \quad I_c[g]\big|_{\partial D} = g \,,$$

- Taylor expanding μ, σ to levels p, q, we get an approximation of u^{n+1}:

$$u^{n+1,p,q} = I_c[u_0] + \sum_{k=0}^{p} \frac{\mu^{(k)}(0)}{k!} I[(u^{n,p,q})^k] + \sum_{k=0}^{q} \frac{\sigma^{(k)}(0)}{k!} I[(u^{n,p,q})^k \xi] \,.$$

- $u^{n,p,q}$ are part of model $\mathcal{M}$ built from $\mathcal{M}^0 = \{I_c[u_0]\}$ and ξ.
- As $p, q \to \infty$, we expect $u^{n,p,q} \to u^n$; as $n \to \infty$, we expect $u^n \to u$.
- $\Rightarrow$ for every $x \in D$, linear combinations of $\{f(x)\}_{f\in\mathcal{M}}$ should well-approximate $u(x)$.

Numerical experiments

Parabolic PDE with forcing

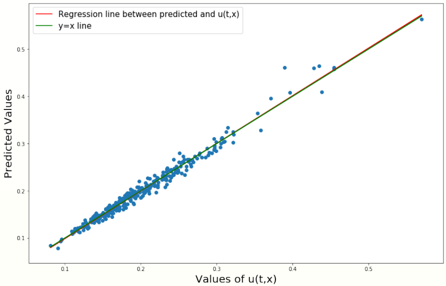
For input $\xi\colon [0,1] \times [0,1] \to \mathbb{R}$, consider

$$\begin{aligned}(\partial_t - \partial_x^2)u &= 3u - u^3 + u\,\xi \quad \text{on } [0,1] \times [0,1],\\ u(t,0) &= u(t,1) \quad \text{(Periodic BC)},\\ u(0,x) &= x(1-x)\,.\end{aligned}$$

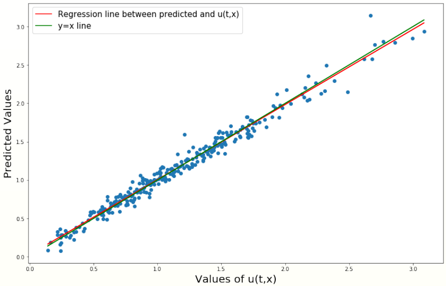
Aim: for fixed $(t,x) \in [0,1] \times [0,1]$, learn $u(t,x)$ from ξ by linear regression at against model at (t,x).

Method:

- Sample 1000 realisations of ξ as white noise.
- Train/test split: 700/300. On training set, solve the PDE numerically.
- On train and test sets, compute the models $\{f\}_{f\in\mathcal{M}}$ with $|\mathcal{M}| < 60$ functions.
- Here: $I = (\partial_t - \partial_x^2)^{-1}$ and $\mathcal{M}^0 = \emptyset$ ('forget' the initial condition)
- Fit linear regression of $u(t,x)$ against $\{f(t,x)\}_{f\in\mathcal{M}}$ from training set.
- Apply fit on testing set.



(a) Prediction at $(t, x) = (0.05, 0.5)$.
Relative ℓ^2 error: 4.7%. Slope: 1.01.



(b) Prediction at $(t, x) = (1, 0.5)$.
Relative ℓ^2 error: 6.9%. Slope: 0.98.

	(t, x) = (0.05, 0.5)		(t, x) = (0.5, 0.5)		(t, x) = (1, 0.5)		(t, x) = (1, 0.95)	
Model's Height	Error	Slope	Error	Slope	Error	Slope	Error	Slope
1	8.83%	0.91	21.14%	0.85	22.81%	0.72	22.95%	0.75
2	5.60%	0.96	9.79%	0.97	13.42%	0.91	13.16%	0.91
3	5.15%	0.97	8.15%	0.98	7.90%	0.97	8.69%	0.96
4	4.88%	0.97	7.85%	0.98	6.61%	0.98	7.06%	0.98

Remark: similar for additive forcing, but prediction worsens far from boundary.

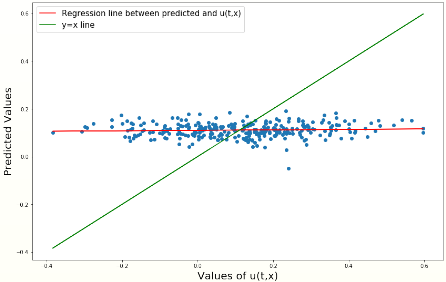
Wave equation with forcing

As before, but for wave equation

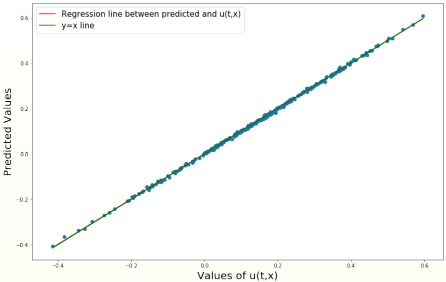
$$
\begin{aligned}
(\partial_t^2 - \partial_x^2)u &= \cos(\pi u) + u^2 + u\,\xi \quad \text{for } (t,x) \in [0,1]\times[0,1],\\
u(t,0) &= u(t,1) \quad \text{(Periodic BC)},\\
u(0,x) &= u_0(x) := \sin(2\pi x),\\
\partial_t u(0,x) &= v_0(x) := x(1-x)\,,
\end{aligned}
$$

- **Aim:** for fixed $(t,x) \in [0,1]\times[0,1]$, learn $u(t,x)$ from ξ by linear regression at against model at (t,x).
- Now $I = (\partial_t^2 - \partial_x^2)^{-1}$ and include **both** initial condition and speed in initialising set, $\mathcal{M}^0 = \{I_c[u_0], I_s[v_0]\}$:

$$
\begin{cases}
(\partial_t^2 - \partial_x^2)I_c[u_0] & = \ 0\\
I_c[u_0](0,x) & = u_0(x)\,,\\
\partial_t I_c[u_0](0,x) & = 0\,.
\end{cases}
\qquad
\begin{cases}
(\partial_t^2 - \partial_x^2)I_s[v_0] & = \ 0\\
I_s[v_0](0,x) & = 0\,,\\
\partial_t I_s[v_0](0,x) & = v_0(x)\,.
\end{cases}
$$



(a) Prediction at $(t, x) = (1, 0.5)$ for model with $\mathcal{M}^0 = \emptyset$. Relative ℓ^2 error: 84.1%.



(b) Prediction at $(t, x) = (1, 0.5)$ for model with $\mathcal{M}^0 = \{I_c[u_0], I_s[v_0]\}$. Relative ℓ^2 error: 1.8%.

Model's Height	1	2	3	4
With initial speed	60.60%	12.86%	2.09%	1.19%
Without initial speed	60.40%	13.45%	5.39%	4.77%

Burgers' equation

Final experiment: learn *entire* solution $\{u(t,x)\}_{(t,x)\in[0,10]\times[-8,8]}$ of

$$
\begin{gathered}
(\partial_t - 0.1\partial_x^2)u = -u\partial_x u \quad (t,x) \in [0,10] \times [-8,8] \\
u(t,-8) = u(t,8) \quad \text{(Periodic BC)}\,, \\
u_0(x) = \sum_{k=-10}^{10} \frac{a_k}{1+|k|^2} \sin\left(\lambda^{-1}\pi k x\right)
\end{gathered}
$$

- Input: initial condition u_0 — $(a_k)_{k=-10,\ldots,10}$ i.i.d. standard normal, $\lambda = 2,4,8$ uniformly.
- Train/test split: 100/20. On training set, solve PDE numerically.

Burgers' equation

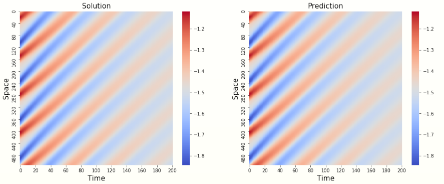
- **No forcing** $\Rightarrow$ learn dynamical system: find functions $a, b\colon [-8,8] \to \mathbb{R}$ such that, for some $\delta > 0$ and all $k = 0, \ldots, 10/\delta$,

$$u((k+1)\delta, \cdot) \approx a(\cdot) + \sum_{f \in \mathcal{M}} b_f(\cdot) f(\delta, \cdot)\,,$$

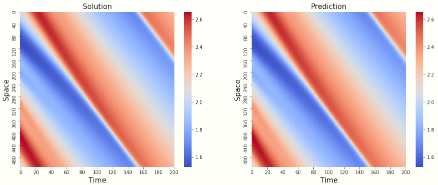
  where $\mathcal{M}$ is model as in heat equation but on $[0,\delta] \times [-8,8]$ and with $\xi \equiv 0$ and initialising set $\mathcal{M}^0 = \{I_c[u(k\delta, \cdot)]\}$.

- We divide $[0,10]$ into 200 intervals of length $\delta = 0.05$.
- On training set, fit a linear regression for functions $a(x), b_f(x)$ at each $x \in [-8,8]$ (constant in time!)
- $\Rightarrow$ training set size effectively increases $100 \rightsquigarrow 200 \times 100$.
- **Result:** ℓ^2 error average: 3.04%, range: $\approx 0\%$ to 11.4% (over 10 experiment repeats).

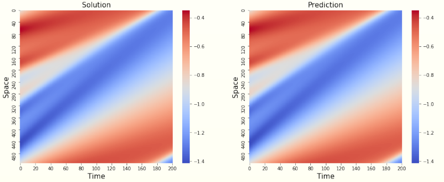
Heat-maps for true and predicted solutions from four test cases.



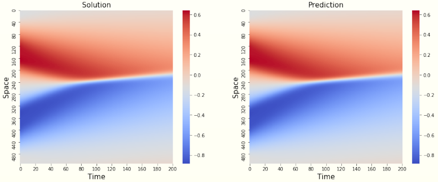
(a) Relative ℓ^2 error: 0.6%.



(b) Relative ℓ^2 error: 1.4%.



(c) Relative ℓ^2 error: 2.4%.



(d) Relative ℓ^2 error: 7.9%.

Remarks – Burgers' equation experiment

- Predictive power stable under noisy observations.
- The viscosity $\nu = 0.1$ in PDE can be estimated.
- Benchmarked against two other methods:
  - Naive Euler regression algorithm: much less predictive power
  - An adaptation of PDE-FIND algorithm[4] to learn **coefficients of PDE**: almost as good on original data, but much worse on noisy data.

[4] Samuel H Rudy et al. "Data-driven discovery of partial differential equations". *Science Advances* 3.4 (2017), e1602614.

Further directions

- Applications beyond PDEs? Possible domains:
  - meteorological data,
  - image and remote sensing recognition,
  - fluid dynamics.
- Universality properties?
- How to choose 'hyperparameter' l? Can it be learnt?
- Combine with other learning algorithms (neural networks, random forests, etc.)? Kernelisation?

Thank you!