Feature Engineering with Regularity Structures

Ilya Chevyrev

(Joint work with Andris Gerasimovičs & Hendrik Weber)
arXiv:2108.05879

The University of Edinburgh

7 September 2022

BIRS Workshop: New interfaces of Stochastic Analysis and Rough Paths

Overview

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

Background - signatures

Machine learning

Simplistic picture:

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

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

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

Considerations: descriptiveness, vectorisation, dimensional reduction, etc.

Motivating problem: from observed samples of ξ and u, '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'.
  - number and location of 'observed points' can vary from sample to sample
  - feature vectors $\{\xi(x_i)\}_{i=1}^N$ can 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

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

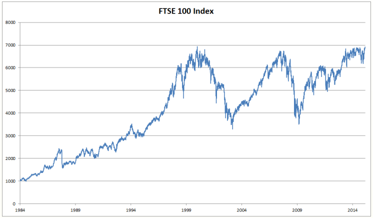
Chen, Ree, Magnus 50's, Brockett, Sussmann, Fliess 70's+, Lyons '90's+

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.

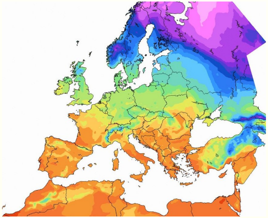
Grandjean 2014 *Les Cahiers du Numérique*

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

- Image recognition.

RSSCN7 dataset [Zou et al. 2015]

- Meteorological data.



ECMWF 2011

Higher dimensions

Generalise signatures to higher dimensions?

- Zhang–Lin–Tindel[1] generalise signatures via spatial differentials.
    - Applied to image and texture classification.
- Giusti–Lee–Nanda–Oberhauser[2] generalise signatures via cubical mapping spaces.

Our approach is based on regularity structures.

- SPDEs: rough paths $\rightsquigarrow$ regularity structures; signatures $\rightsquigarrow$ models

[1] Sheng Zhang, Guang Lin, and Samy Tindel. "2-d signature of images and texture classification". *arXiv e-prints*, arXiv:2205.11236 (May 2022).

[2] Chad Giusti et al. "A Topological Approach to Mapping Space Signatures". *arXiv e-prints*, arXiv:2202.00491 (Feb. 2022).

Motivation – Picard's theorem

Given $\xi\colon D \to \mathbb{R}$, 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 polynomials.

Motivation – Picard's theorem

Given $\xi\colon D \to \mathbb{R}$, 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 polynomials.
- **Operator:** $I = \mathcal{L}^{-1}$: $\mathcal{L}I[f] = f\,, \quad I[f]\big|_{\partial D} = 0.$
- **Picard's theorem:** $u = \lim_{n\to\infty} u^{(n)}$ where

$$\begin{cases} \mathcal{L}u^{(1)} & = 0\,, \qquad u^{(1)}\Big|_{\partial D} = u_0\,, \\ u^{(n+1)} & = u^{(1)} + I[\mu(u^{(n)})] + I[\sigma(u^{(n)})\xi]\,, \end{cases}$$

Motivation – Picard's theorem

Given $\xi\colon D \to \mathbb{R}$, 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 polynomials.
- **Operator:** $I = \mathcal{L}^{-1}\colon\ \mathcal{L}I[f] = f\,, \quad I[f]\big|_{\partial D} = 0.$
- **Picard's theorem:** $u = \lim_{n\to\infty} u^{(n)}$ where

$$\begin{cases} \mathcal{L}u^{(1)} &= 0\,, \qquad u^{(1)}\Big|_{\partial D} = u_0\,, \\ u^{(n+1)} &= u^{(1)} + I[\mu(u^{(n)})] + I[\sigma(u^{(n)})\xi]\,, \end{cases}$$

- $u^{(n)}$ is multi-linear function of $u^{(1)}$ and ξ.
  $\rightsquigarrow$ **model feature vector**

Models

Definition (Model feature vector)

Static objects: set $D \subset \mathbb{R}^d$, linear operator I acting on $\mathbb{R}^D$.

(Think: $I[u] = K * u$ for a kernel K.)

Input: $(\{u^{(i)}\}_{i=1}^{\ell}, \xi)$ functions $\xi, u^{(i)} \colon D \to \mathbb{R}$.

Models

Definition (Model feature vector)

Static objects: set $D \subset \mathbb{R}^d$, linear operator I acting on $\mathbb{R}^D$.

(Think: $I[u] = K * u$ for a kernel K.)

Input: $(\{u^{(i)}\}_{i=1}^{\ell}, \xi)$ functions $\xi, u^{(i)} \colon D \to \mathbb{R}$.

The **model feature vector** is the family of functions $\cup_{n\geq 0}\mathcal{M}^n$

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

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

(ξ is called forcing.)

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

Signature

Example (Signature)

- Forcing: $\xi\colon [0,T] \to \mathbb{R}$.
- **Operator:** $I_t[\xi] = \int_0^t \xi_s \,\mathrm{d}s$.
- Initialising set: $\mathcal{M}^0 = \emptyset$.
- $\Rightarrow$ functions in **model feature vector** $\mathcal{M}$ evaluated at T encode the **signature** of $X := \int_0^\cdot \xi_s \,\mathrm{d}s$.
  - (Works also for $\xi\colon [0,T] \to \mathbb{R}^n$.)

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).

Parabolic PDE with forcing

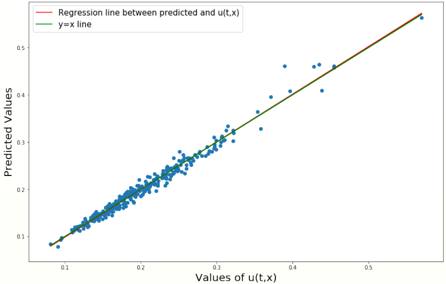
For input $\xi\colon [0,1]\times[0,1]\to\mathbb{R}$, consider

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

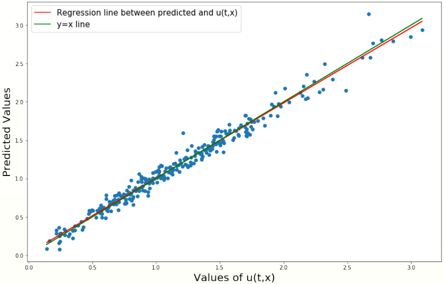
Aim: for fixed $(t,x)\in[0,1]\times[0,1]$, learn $u(t,x)$ from ξ by linear regression at against model at (t,x).

Method:

- Train and test: compute 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)
- **Train:** fit linear regression of $u(t,x)$ against $\{f(t,x)\}_{f\in\mathcal{M}}$.
- **Test:** apply fit from training step.



(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).

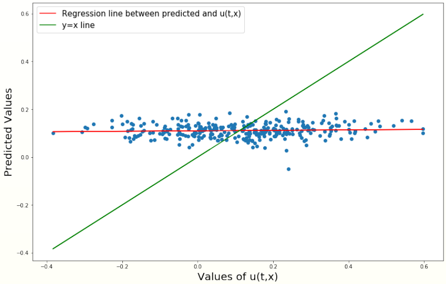
Wave equation with forcing

As before, but for wave equation

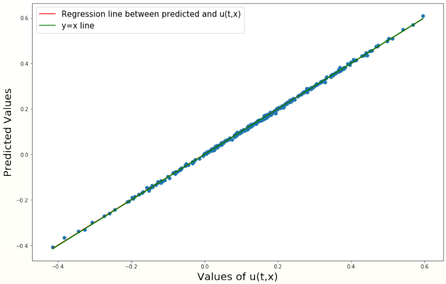
$$
\begin{aligned}
(\partial_t^2 - \partial_x^2)u &= \cos(\pi u) + u^2 + u\,\xi \quad \text{for } (t,x) \in [0,1] \times [0,1],\\
u(t,0) &= u(t,1) \quad \text{(Periodic BC)},\\
u(0,x) &= u_0(x) := \sin(2\pi x),\\
\partial_t u(0,x) &= v_0(x) := x(1-x)\,,
\end{aligned}
$$

- **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%

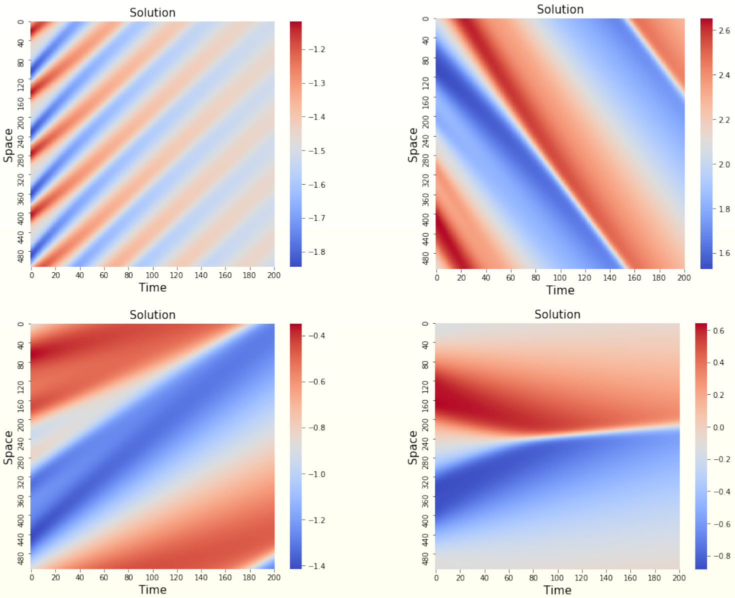
Burgers' equation

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

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

- Input: initial condition u_0 with $(a_k)_{k=-10,\ldots,10}$ i.i.d. standard normal, $\lambda = 2, 4, 8$ uniformly.

Heat-maps for four tests.



Burgers' equation

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

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

  where $\mathcal{M}$ is model as in heat equation but on $[0, \delta] \times [-8, 8]$ and with $\xi \equiv 0$ and initialising set $\mathcal{M}^0 = \{I_c[u(k\delta, \cdot)]\}$.
- Divide $[0, 10]$ into 200 intervals of length $\delta = 0.05$.
- **Train:** 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 $100 \rightsquigarrow 200 \times 100$.

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[3] to learn **coefficients of PDE**: almost as good on original data, but much worse on noisy data.

[3] 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?
  - Recently combined with neural networks by Hu et al.[4]
  - See also Salvi–Lemercier–Gerasimovics.[5]

[4] Peiyan Hu et al. "Neural Operator with Regularity Structure for Modeling Dynamics Driven by SPDEs". *arXiv e-prints*, arXiv:2204.06255 (Apr. 2022).

[5] Cristopher Salvi, Maud Lemercier, and Andris Gerasimovics. "Neural Stochastic Partial Differential Equations: Resolution-Invariant Learning of Continuous Spatiotemporal Dynamics". *arXiv e-prints*, arXiv:2110.10249 (Oct. 2021).

Thank you!