Learning solution operators for PDEs with uncertainty

PN Springschool 2023

Emilia Magnani

29 March 2023

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

How do we solve a PDE?

Operator learning problem

Let $\mathcal{L}_\lambda$ be a differential operator depending on a parameter $\lambda \in \mathcal{A}$.

We want to find the solution $u(x)$ of the PDE

$$\begin{aligned} &\left(\mathcal{L}_\lambda u\right)(x) = f(x) && x \in D \subset \mathbb{R}^d \\ &\text{boundary condition} && x \in \partial D \end{aligned} \tag{1}$$

How do we solve a PDE?

Operator learning problem

Let $\mathcal{L}_\lambda$ be a differential operator depending on a parameter $\lambda \in \mathcal{A}$.

We want to find the solution $u(x)$ of the PDE

$$\begin{aligned} (\mathcal{L}_\lambda u)(x) &= f(x) && x \in D \subset \mathbb{R}^d \\ \text{boundary condition} && x \in \partial D \end{aligned} \tag{1}$$

Case 0. λ, f *fixed*. $\rightarrow$ Numerical methods (FEM, FD...)

How do we solve a PDE?

Operator learning problem

Let $\mathcal{L}_\lambda$ be a differential operator depending on a parameter $\lambda \in \mathcal{A}$.

We want to find the solution $u(x)$ of the PDE

$$\begin{aligned} (\mathcal{L}_\lambda u)(x) &= f(x) && x \in D \subset \mathbb{R}^d \\ \text{boundary condition} && x \in \partial D \end{aligned} \tag{1}$$

Case 0. λ, f *fixed*. $\rightarrow$ Numerical methods (FEM, FD...)

Equation (1) defines a solution operator

$$\mathcal{H} : \Lambda \times \mathcal{F} \longrightarrow \mathcal{U}$$
$$(\lambda, f) \longmapsto u_{\lambda,f} \quad \text{solution of (1)}$$

Case 1. λ *fixed*, f varies. $\rightarrow \quad \mathcal{G}: \ f \mapsto u$

Case 1. λ *fixed*, f varies. $\rightarrow \quad \mathcal{G}: \ f \mapsto u$

If the differential operator $\mathcal{L}_\lambda$ is *linear*, and a Green's function $G : DxD \rightarrow \mathbb{R}$ exists for the operator $\mathcal{L}_\lambda$, we can express the solution $u(x)$ of $\left(\mathcal{L}_\lambda u\right)(x) = f(x)$ as

$$u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

Case 1. λ *fixed*, f varies. $\rightarrow \quad \mathcal{G}: \ f \mapsto u$

If the differential operator $\mathcal{L}_\lambda$ is *linear*, and a Green's function $G : DxD \rightarrow \mathbb{R}$ exists for the operator $\mathcal{L}_\lambda$, we can express the solution $u(x)$ of $\left(\mathcal{L}_\lambda u\right)(x) = f(x)$ as

$$u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

Where

$$\mathcal{L}_\lambda G(x, \cdot) = \delta(\cdot - x) \quad \text{for } x \in D$$

Case 1. λ *fixed*, f varies. $\rightarrow \quad \mathcal{G}: \ f \mapsto u$

If the differential operator $\mathcal{L}_\lambda$ is *linear*, and a Green's function $G : DxD \rightarrow \mathbb{R}$ exists for the operator $\mathcal{L}_\lambda$, we can express the solution $u(x)$ of $(\mathcal{L}_\lambda u)(x) = f(x)$ as

$$u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

Where

$$\mathcal{L}_\lambda G(x, \cdot) = \delta(\cdot - x) \quad \text{for } x \in D$$

We consider the operator

$$\mathcal{G}: \mathcal{F} \longrightarrow \mathcal{U}$$

$$f \longmapsto u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

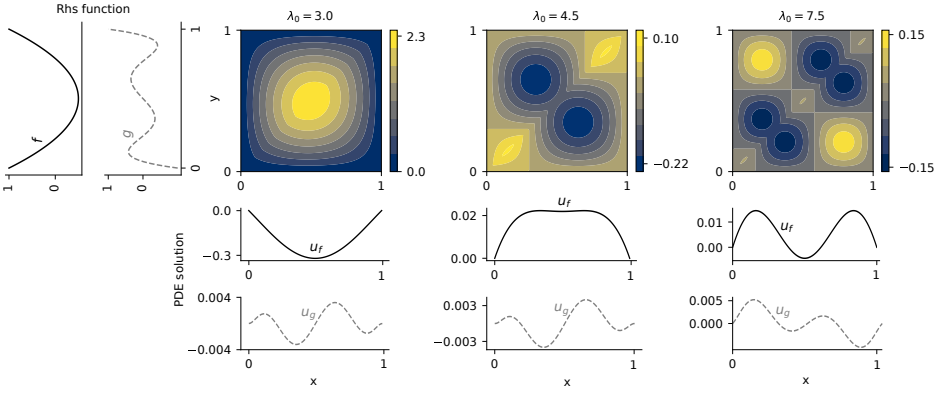
Note that $\mathcal{G} = \mathcal{L}_\lambda^{-1}$ Inverse problem

Linear PDEs and Green's function

A visualization

Example: $(\mathcal{L}_\lambda u)(x) = -\left(\frac{d}{dx^2} + \lambda\right) u(x) = f(x) \quad x \in [0,1], \quad u(0) = u(1) = 0$

with Green's function $G_\lambda(x,y) = \frac{1}{\lambda \sin \lambda}\left[\Theta(y-x)\sin(\lambda x)\sin(\lambda(1-y)) + \Theta(x-y)\sin(\lambda(1-x))\sin(\lambda y)\right]$



Learning the solution operator

i.e. inferring the Green's function

Solution operator ($\lambda \in \mathbb{R}$ fixed)

$$\mathcal{G}: \mathcal{F} \longrightarrow \mathcal{U}$$

$$f \longmapsto u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

Learning the solution operator

i.e. inferring the Green's function

Solution operator ($\lambda \in \mathbb{R}$ fixed)

$$\mathcal{G} : \mathcal{F} \longrightarrow \mathcal{U}$$

$$f \longmapsto u(x) = \int_D G_\lambda(x, y)\, f(y)\, dy$$

Inferring the solution operator $\mathcal{G} : f \mapsto u$ equivalent to learning the function $G : \mathbb{R}^2 \to \mathbb{R}$.

Learning the solution operator

i.e. inferring the Green's function

Solution operator ($\lambda \in \mathbb{R}$ fixed)

$$\mathcal{G}: \mathcal{F} \longrightarrow \mathcal{U}$$

$$f \longmapsto u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

Inferring the solution operator $\mathcal{G}: f \mapsto u$ equivalent to learning the function $G : \mathbb{R}^2 \to \mathbb{R}$.

Data points $\{f_i, u_i\}_{i=1}^N$ $\quad u_i = \int_D G(x,y) f_i(y) dy$ (Integral observations)

.

Learning the solution operator

i.e. inferring the Green's function

Solution operator ($\lambda \in \mathbb{R}$ fixed)

$$\mathcal{G}: \mathcal{F} \longrightarrow \mathcal{U}$$

$$f \longmapsto u(x) = \int_D G_\lambda(x,y)\, f(y)\, dy$$

Inferring the solution operator $\mathcal{G}: f \mapsto u$ equivalent to learning the function $G : \mathbb{R}^2 \to \mathbb{R}$.

Data points $\{f_i, u_i\}_{i=1}^N$ $\quad u_i = \int_D G(x,y) f_i(y) dy$ (Integral observations)

.

We want to learn $\mathcal{G}$ in a probabilistic framework.

Gaussian Process regression

with Integral observations

Define the integral operator $\mathcal{A}_f = \mathcal{A}$ acting on G as $\mathcal{A}G = \int_D G(\cdot, y)f(y)dy = u(\cdot)$.

Gaussian Process regression

with Integral observations

Define the integral operator $\mathcal{A}_f = \mathcal{A}$ acting on G as $\mathcal{A}G = \int_D G(\cdot, y)f(y)dy = u(\cdot)$.

Prior $G \sim \mathcal{GP}(\mu, k_\theta)$ with mean $\mu \colon \mathbb{R}^2 \to \mathbb{R}$ and $k_\theta \colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$

Likelihood $u \mid G \sim \mathcal{N}(\mathcal{A}G, \sigma^2)$

Gaussian Process regression

with Integral observations

Define the integral operator $\mathcal{A}_f = \mathcal{A}$ acting on G as $\mathcal{A}G = \int_D G(\cdot, y)f(y)dy = u(\cdot)$.

Prior $G \sim \mathcal{GP}(\mu, k_\theta)$ with mean $\mu\colon \mathbb{R}^2 \to \mathbb{R}$ and $k_\theta\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$

Likelihood $u \mid G \sim \mathcal{N}(\mathcal{A}G, \sigma^2)$

Posterior mean and covariance

$$\mathbb{E}[G] = \mu + \mathcal{A}^* k_\theta (\mathcal{A}\mathcal{A}^* k_\theta + \sigma^2)^{-1}(u - \mathcal{A}\mu)$$

$$\mathrm{Cov}(G) = k_\theta - \mathcal{A}^* k_\theta (\mathcal{A}\mathcal{A}^* k_\theta + \sigma^2)^{-1}\mathcal{A}k_\theta.$$

where

$$\mathcal{A}^* k_\theta = \int_D k_\theta((x, y), (X, \tau))f(\tau)d\tau, \quad \mathcal{A}\mathcal{A}^* k_\theta = \int_D k_\theta((X, \tau_1), (X, \tau_2))f(\tau_1)f(\tau_2)d\tau_1 d\tau_2$$

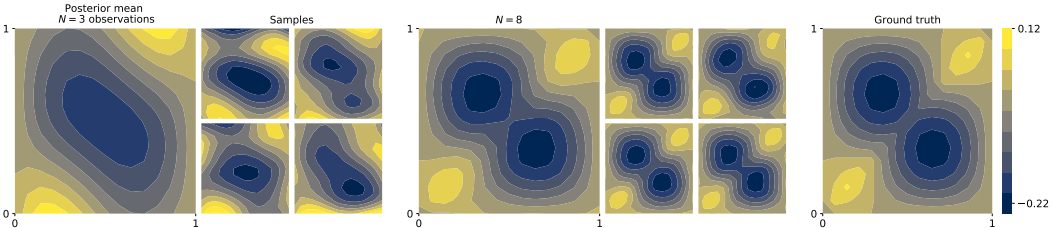
(when discretized $\mathcal{A}\mathcal{A}^* k_\theta = A k_\theta A^T$)

An experiment

Posterior distribution on the operator

$$(\mathcal{L}_\lambda u)(x) = (-\Delta - \lambda^2) u(x) = f(x) \quad x \in [0,1], \quad \lambda = 4.5 \quad u(0) = u(1) = 0$$

Data $\{f_i, u_i\}_{i=1}^N$, Product kernel $k((x,x'),(y,y')) = k_1(x,y)\, k_2(x',y')$



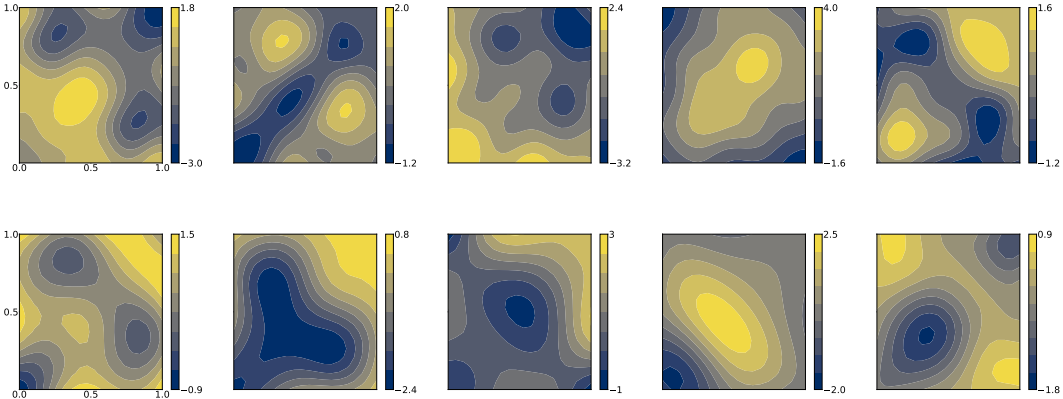
Posterior on G and ground truth [Magnani et al. 2022]

Encoding symmetries in the kernel

Prior knowledge

We can encode prior knowledge about the Green's function in the choice of our kernel.

For the Green's function $G(x_1, x_2) = G(x_2, x_1)$.



Deep neural networks to solve PDEs

Neural operators, FNO ... [Lu et al 2021] [Li et al 2020,2021] [Kowachki et al 2021]

Case 2. λ, f vary.

Deep neural networks to solve PDEs

Neural operators, FNO ... [Lu et al 2021] [Li et al 2020,2021] [Kowachki et al 2021]

Case 2. λ, f vary.

Approximating the solution operator

$$\mathcal{H}: \Lambda \times \mathcal{F} \longrightarrow \mathcal{U}$$

$$(\lambda, f) \longmapsto u_{\lambda,f} \quad \text{solution of } \mathcal{L}_\lambda u = f$$

Deep neural networks to solve PDEs

Neural operators, FNO ... [Lu et al 2021] [Li et al 2020,2021] [Kowachki et al 2021]

Case 2. λ, f vary.

Approximating the solution operator

$$\mathcal{H}: \Lambda \times \mathcal{F} \longrightarrow \mathcal{U}$$

$$(\lambda, f) \longmapsto u_{\lambda,f} \quad \text{solution of } \mathcal{L}_\lambda u = f$$

through the composition of L layers $(\lambda, f) \longmapsto (\psi_L \circ \cdots \circ \psi_1)(\lambda, f)$

An iterative solver $\underbrace{\psi_{i+1}(f,\lambda) = \sigma\left(W_i\psi_i(x) + \int_D \overbrace{g_\theta(x,y,\lambda)}^{\text{NN1}}\, \psi_i(y)\, dy\right)}_{\text{NN2}}$ $\qquad i = 0, \dots N$

$\rightarrow$ **Case 1** gives a theoretical understanding to these complex architectures

Deep neural networks to solve PDEs

Neural operators, FNO ... [Lu et al 2021] [Li et al 2020,2021] [Kowachki et al 2021]

Case 2. λ, f vary.

Approximating the solution operator

$$\mathcal{H}: \Lambda \times \mathcal{F} \longrightarrow \mathcal{U}$$

$$(\lambda, f) \longmapsto u_{\lambda,f} \quad \text{solution of } \mathcal{L}_\lambda u = f$$

through the composition of L layers $(\lambda, f) \longmapsto (\psi_L \circ \cdots \circ \psi_1)(\lambda, f)$

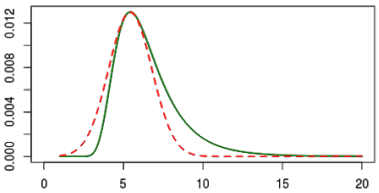
An iterative solver $\underbrace{\psi_{i+1}(f,\lambda) = \sigma\left(W_i\psi_i(x) + \int_D \overbrace{g_\theta(x,y,\lambda)}^{\text{NN1}}\, \psi_i(y)\, dy\right)}_{\text{NN2}}$ $\qquad i = 0, \dots N$

→ **Case 1** gives a theoretical understanding to these complex architectures

Uncertainty becomes even more important

Uncertainty with Laplace approximation (LA)

Old but gold



$q(\theta) = \log h(\theta)$ Taylor expansion of $q(\theta)$ around the mode $\hat{\theta}$:

$$q(\theta) \approx q(\hat{\theta}) + \overbrace{q'(\hat{\theta})(\theta - \hat{\theta})}^{=0} + \frac{1}{2}(\theta - \hat{\theta})q''(\hat{\theta})(\theta - \hat{\theta})$$

$$= c - \frac{(\theta - \mu)^2}{2\sigma^2} \rightarrow \log \mathcal{N}(\mu, \sigma^2), \quad \mu = \hat{\theta}, \quad \sigma^2 = (-q(\hat{\theta})'')^{-1}$$

Laplace approximation in Deep learning

[MacKay 1992], [Daxberger, Kristiadi, Immer, Eschenhagen, Bauer, Hennig 2021]

Consider a deep network $f_\theta : \mathbb{R}^m \to \mathbb{R}^n$ trained on $\mathcal{D} = (x_i \in \mathbb{R}^m, y_i \in \mathbb{R}^n)_{i=1}^N$ to find

$$\theta_{\text{MAP}} = \arg\min_\theta \mathcal{L}(\mathcal{D};\theta) = \arg\min_\theta \left(\frac{1}{N} \sum_1^N L(x_n, y_n; \theta) + r(\theta) \right)$$

$$= \arg\max_\theta - \Big(\log p(y|f_\theta(x)) + \log p(\theta) \Big) = \arg\max_\theta p(\theta|y)$$

$$\boxed{p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_{\text{MAP}}, \Sigma) \quad \text{with } \Sigma = -(\nabla^2_\theta \mathcal{L}(\mathcal{D};\theta))^{-1}}$$

Laplace approximation in Deep learning

[MacKay 1992], [Daxberger, Kristiadi, Immer, Eschenhagen, Bauer, Hennig 2021]

Consider a deep network $f_\theta : \mathbb{R}^m \to \mathbb{R}^n$ trained on $\mathcal{D} = (x_i \in \mathbb{R}^m, y_i \in \mathbb{R}^n)_{i=1}^N$ to find

$$\theta_{\text{MAP}} = \arg\min_\theta \mathcal{L}(\mathcal{D};\theta) = \arg\min_\theta \left(\frac{1}{N} \sum_1^N L(x_n, y_n; \theta) + r(\theta) \right)$$

$$= \arg\max_\theta - \Big(\log p(y|f_\theta(x)) + \log p(\theta) \Big) = \arg\max_\theta p(\theta|y)$$

$$\boxed{p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_{\text{MAP}}, \Sigma) \quad \text{with } \Sigma = -(\nabla^2_\theta \mathcal{L}(\mathcal{D};\theta))^{-1}}$$

..and a gaussian distribution over the (linearized) network output

$$p(f(x) \mid x, D) \approx \mathcal{GP}\big(f;\ f_{\theta_{\text{MAP}}}(x),\ J(x)^T \Sigma J(x)\big) \quad \text{with } J = \nabla_\theta f_\theta(x)|_{\theta_{\text{MAP}}}$$

Laplace approximation in Deep learning

[MacKay 1992], [Daxberger, Kristiadi, Immer, Eschenhagen, Bauer, Hennig 2021]

Consider a deep network $f_\theta : \mathbb{R}^m \to \mathbb{R}^n$ trained on $\mathcal{D} = (x_i \in \mathbb{R}^m, y_i \in \mathbb{R}^n)_{i=1}^N$ to find

$$\theta_{\text{MAP}} = \arg\min_\theta \mathcal{L}(\mathcal{D};\theta) = \arg\min_\theta \left(\frac{1}{N} \sum_1^N L(x_n, y_n; \theta) + r(\theta) \right)$$

$$= \arg\max_\theta - \Big(\log p(y|f_\theta(x)) + \log p(\theta) \Big) = \arg\max_\theta p(\theta|y)$$

$$\boxed{p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_{\text{MAP}}, \Sigma) \quad \text{with } \Sigma = -(\nabla^2_\theta \mathcal{L}(\mathcal{D};\theta))^{-1}}$$

..and a gaussian distribution over the (linearized) network output

$$p(f(x) \mid x, D) \approx \mathcal{GP}\big(f;\ f_{\theta_{\text{MAP}}}(x),\ J(x)^T \Sigma J(x)\big) \quad \text{with } J = \nabla_\theta f_\theta(x)|_{\theta_{\text{MAP}}}$$

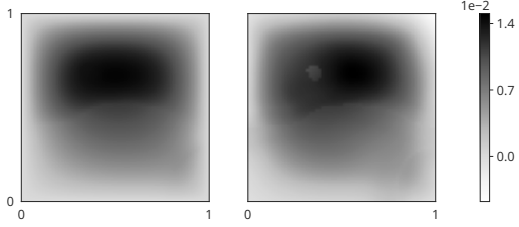
$\to$ Small computational overhead!

Uncertainty estimates on a Darcy flow

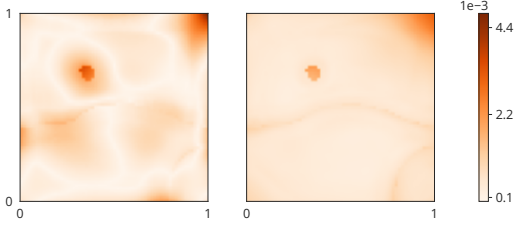
[Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, Anandkumar, 2020]

Darcy flow in $D = [0,1]^2$

$$-\nabla \cdot (\lambda(x)\nabla u(x)) = f(x), \quad x \in D$$
$$u(x) = 0 \qquad x \in \partial D$$



Ground truth and approximation



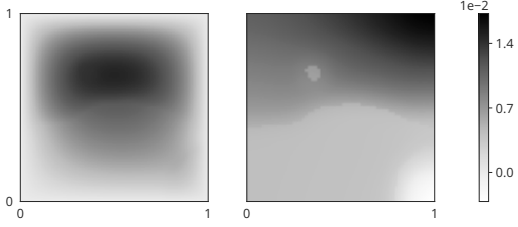
Error and standard deviation [Magnani et al. 2022]

Uncertainty estimates on Darcy flow

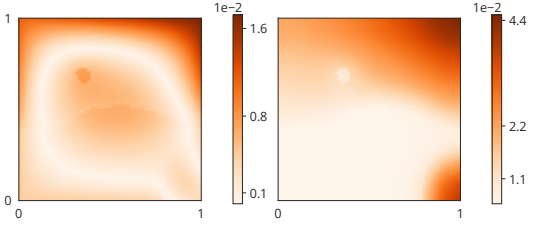
Low-data regime

Darcy flow in $D = [0,1]^2$

$$-\nabla \cdot (\lambda(x)\nabla u(x)) = f(x), \quad x \in D$$
$$u(x) = 0 \qquad x \in \partial D$$



Ground truth and approximation



Error and standard deviation

Operator learning in the context of inverse problems

What does it mean to learn an operator?

Learned the operator $\mathcal{G}: f \mapsto u$ from data $(f_i, u_i)_{i=1}^N$ (ML setting)

Operator learning in the context of inverse problems

What does it mean to learn an operator?

Learned the operator $\mathcal{G}: f \mapsto u$ from data $(f_i, u_i)_{i=1}^N$ (ML setting)

In general: need assumptions on $\mathcal{G}$!

(Bayesian approach)

- [Trabs, 2018]
- [V.de Hoop et al. 2022]

Operator learning in the context of inverse problems

What does it mean to learn an operator?

Learned the operator $\mathcal{G}: f \mapsto u$ from data $(f_i, u_i)_{i=1}^N$ (ML setting)

In general: need assumptions on $\mathcal{G}$!

(Bayesian approach)

- [Trabs, 2018]
- [V.de Hoop et al. 2022]

Our assumption: integral operator $\mathcal{G} = \int g(x,y)f(y)\,dy$, learn g

Operator learning in the context of inverse problems

What does it mean to learn an operator?

Learned the operator $\mathcal{G}: f \mapsto u$ from data $(f_i, u_i)_{i=1}^N$ (ML setting)

In general: need assumptions on $\mathcal{G}$!

(Bayesian approach)

- [Trabs, 2018]
- [V.de Hoop et al. 2022]

Our assumption: integral operator $\mathcal{G} = \int g(x,y)f(y)\,dy$, learn g

$\rightarrow$ inverse problem

$$u = \mathcal{A}_f g + \varepsilon \qquad \text{with} \qquad \mathcal{A}_f g = \int g(x,y)f(y)\,dy$$

Classical inverse problem setting $u = \mathcal{A}g + \varepsilon$ $\mathcal{A}$ given, $(g_i, u_i)_{i=1}^N$

$\longrightarrow$ learn $\hat{F} =$ "$\mathcal{A}^{-1}$" *stable* such that for a new u_*, $g_* \approx \hat{F}(u_*)$

Convolution operators

A special case, a starting point

Integral operator, for $\mathcal{F}, \mathcal{U}$ Hilbert spaces $\mathcal{G} : \mathcal{F} \to \mathcal{U}$ $\quad f \mapsto u(x) = \int G(x,y) f(y)\, dy$.

Assume that $G(x,y) = g(x-y)$

Integral operator, for $\mathcal{F}, \mathcal{U}$ Hilbert spaces $\mathcal{G} : \mathcal{F} \to \mathcal{U} \quad f \mapsto u(x) = \int G(x,y) f(y)\, dy$.

Assume that $G(x,y) = g(x-y)$

$$\mathcal{G}\, f(x) = g * f = \int g(x-y) f(y)\, dy = \int f(x-y) g(y)\, dy = f * g$$

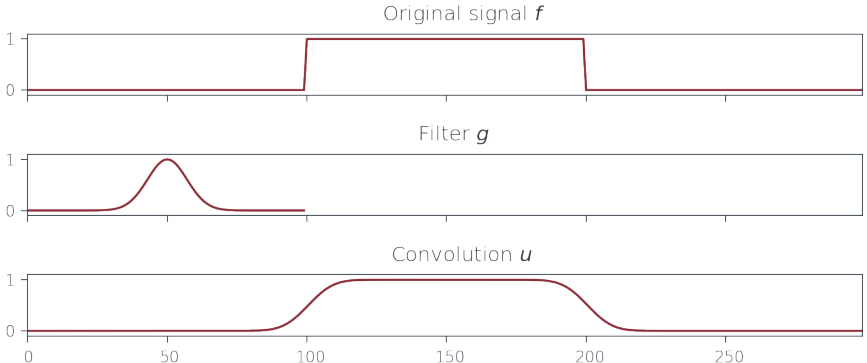
Convolution operators

A special case, a starting point

Integral operator, for $\mathcal{F}, \mathcal{U}$ Hilbert spaces $\mathcal{G} : \mathcal{F} \to \mathcal{U}$ $\quad f \mapsto u(x) = \int G(x,y) f(y)\, dy$.

Assume that $G(x,y) = g(x-y)$

$$\mathcal{G} f(x) = g * f = \int g(x-y) f(y)\, dy = \int f(x-y) g(y)\, dy = f * g$$



Convolution operators

A special case, a starting point

Integral operator, for $\mathcal{F}, \mathcal{U}$ Hilbert spaces $\mathcal{G} : \mathcal{F} \to \mathcal{U} \quad f \mapsto u(x) = \int G(x,y) f(y)\, dy$.

Assume that $G(x,y) = g(x-y)$

$$\mathcal{G}\, f(x) = g * f = \int g(x-y) f(y)\, dy = \int f(x-y) g(y)\, dy = f * g$$

- Special family of of PDEs
- Learning an operator with structure: known diagonalization! (Fourier basis)
- Convolution are at the core of many applications

Let L be a differential operator

$$L = \sum_{\alpha \leq m} C_\alpha D^\alpha$$

A function $U \in \mathcal{D}(\mathbb{R}^n)$ that satisfies

$$LU = \delta$$

is called fundamental solution of the operator L.

Let L be a differential operator

$$L = \sum_{\alpha \leq m} C_\alpha D^\alpha$$

A function $U \in \mathcal{D}(\mathbb{R}^n)$ that satisfies

$$LU = \delta$$

is called fundamental solution of the operator L.

Given the differential equation

$$Lu = f \tag{2}$$

If U is a fundamental solution of L, then a solution of (2) is given by

$$u = U * f.$$

Let L be a differential operator

$$L = \sum_{\alpha \leq m} C_\alpha D^\alpha$$

A function $U \in \mathcal{D}(\mathbb{R}^n)$ that satisfies

$$LU = \delta$$

is called fundamental solution of the operator L.

Given the differential equation

$$Lu = f \tag{2}$$

If U is a fundamental solution of L, then a solution of (2) is given by

$$u = U * f.$$

Important examples for L are

Laplacian $\Delta_x = \sum_i \frac{\partial^2}{\partial x_i^2}$ Heat operator $\partial_t - \Delta_x$ Wave operator $\partial_{tt} - c^2 \Delta_x$

1d Heat equation

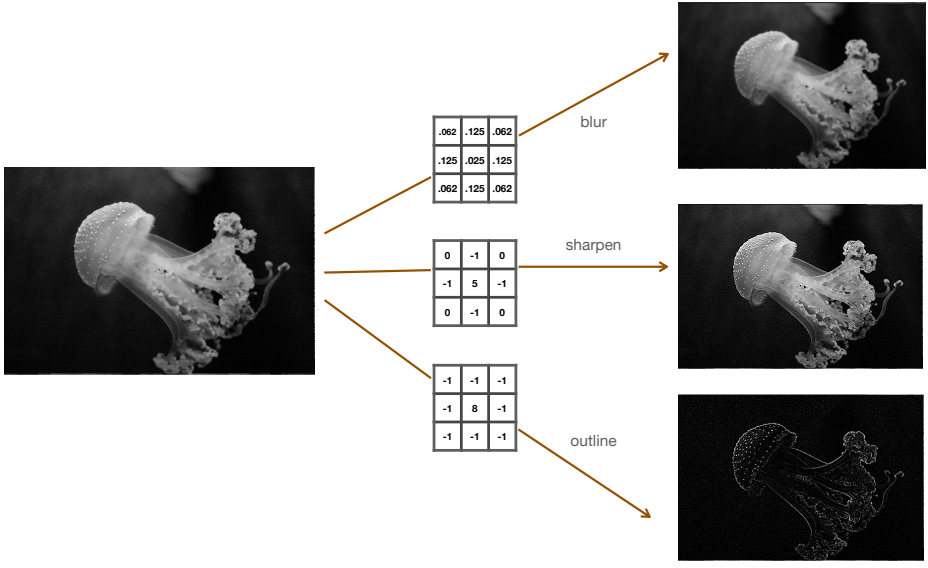
$$\frac{\partial}{\partial t}u(x,t) = \frac{\partial^2}{\partial x^2}u(x,t) \tag{3}$$

Numerical algorithms for solving PDEs (e.g. method of lines) discretize in space on a mesh $D = \{x_1, x_1 + h ... x_1 + dh\}$ and solve in time.
So (3) becomes

$$\frac{\partial}{\partial t}u_d = \frac{K}{h^2}u_d \quad \text{with} \quad K = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & 1 \\ & & & 1 & -2 \end{bmatrix} \quad \text{and} \quad h = \frac{1}{d+1}$$

Second derivative is replaced by second differences at grid points:

$$\frac{\partial^2}{\partial x^2}u(x,t) = \frac{u(x-h,t) - 2u(x,t) + u(x+h,t)}{h^2}$$


.062 .125 .062
.125 .025 .125
.062 .125 .062
blur
0 -1 0
-1 5 -1
0 -1 0
sharpen
-1 -1 -1
-1 8 -1
-1 -1 -1
outline

4. Probability theory

See: 3blue1Brown for a better visualization

The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

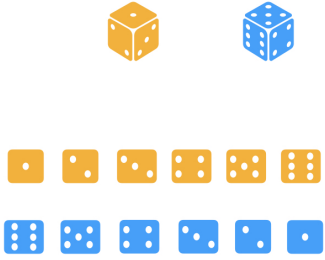
$$Z = X + Y, \quad P(Z = z) = \sum_{k} P(X = k)P(Y = z - k)$$

The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

$$Z = X + Y, \quad P(Z = z) = \sum_{k} P(X = k)P(Y = z - k)$$

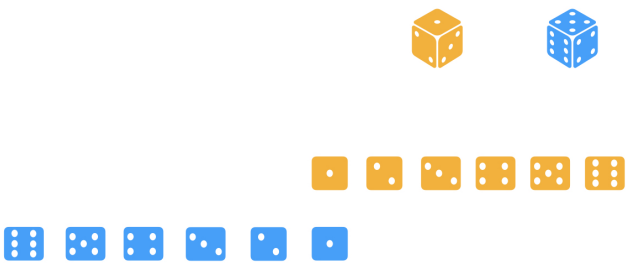
The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

$$Z = X + Y, \quad P(Z = z) = \sum_{k} P(X = k)P(Y = z - k)$$



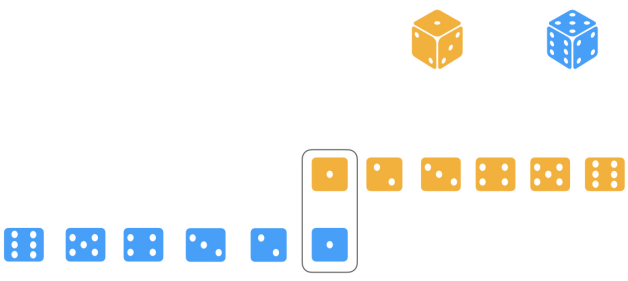
The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

$$Z = X + Y, \quad P(Z = z) = \sum_{k} P(X = k)P(Y = z - k)$$



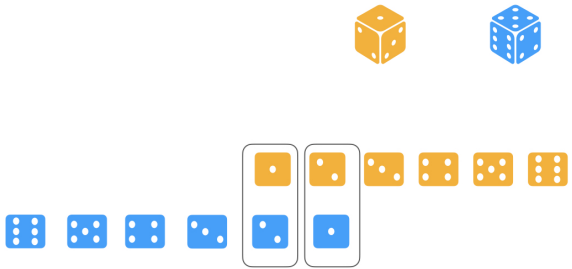
The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

$$Z = X + Y, \quad P(Z = z) = \sum_k P(X = k)P(Y = z - k)$$



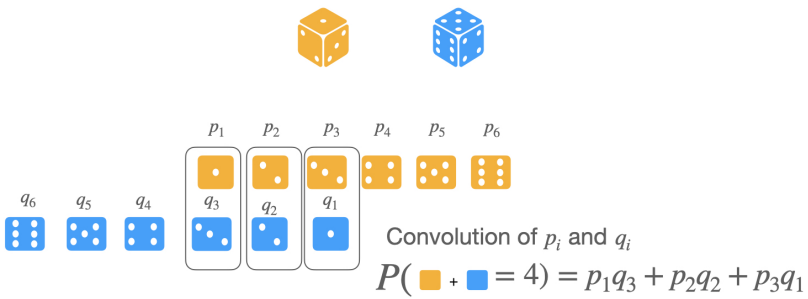
The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

$$Z = X + Y, \quad P(Z = z) = \sum_k P(X = k)P(Y = z - k)$$



The probability distribution of the sum of two independent random variables is the convolution of their individual distributions.

$$Z = X + Y, \quad P(Z = z) = \sum_k P(X = k)P(Y = z - k)$$



So many convolutions!

Operation that shows up in a lot of seemingly unrelated areas.

So many convolutions!

Operation that shows up in a lot of seemingly unrelated areas.

Aim: theoretical investigation of the process of learning convolution operators
in the context of *function spaces*

Kernel ridge regression. $\mathcal{H}$ RKHS

$$\tilde{w} = \arg\min_{w \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \left\| F_w(x_i) - y_i \right\|_{\mathcal{Y}}^2 + \lambda \left\| w \right\|_{\mathcal{H}}^2$$

So many convolutions!

Operation that shows up in a lot of seemingly unrelated areas.

Aim: theoretical investigation of the process of learning convolution operators in the context of *function spaces*

Kernel ridge regression. $\mathcal{H}$ RKHS

$$\tilde{w} = \arg\min_{w \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \left\| F_w(x_i) - y_i \right\|_{\mathcal{Y}}^2 + \lambda \left\| w \right\|_{\mathcal{H}}^2$$

For convolution operators

1. Transposition of the problem: operator valued kernels become tractable
2. Interpretable source conditions

Operator valued kernels

An overview

Scalar case $y_i = f(x_i) + \varepsilon_i, \quad f\colon \mathcal{X} \to \mathcal{Y}, \quad \mathcal{X} = \mathbb{R}^d, \quad \mathcal{Y} = \mathbb{R}$

- $\mathcal{H}$ parameter (Hilbert) space
- Feature map $\phi\colon \mathcal{X} \to \mathcal{H}$, search for $f_w(x) = \langle w, \phi(x) \rangle_{\mathcal{H}} \in \mathcal{Y}, \quad w \in \mathcal{H}$
- $\mathcal{F} = \{f_w | w \in \mathcal{H}\}$ RKHS with kernel $K\colon \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{Y}$, $K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$

[1] See Houman Owhadi

Operator valued kernels

An overview

Scalar case $y_i = f(x_i) + \varepsilon_i, \quad f\colon \mathcal{X} \to \mathcal{Y}, \quad \mathcal{X} = \mathbb{R}^d, \quad \mathcal{Y} = \mathbb{R}$

- $\mathcal{H}$ parameter (Hilbert) space
- Feature map $\phi\colon \mathcal{X} \to \mathcal{H}$, search for $f_w(x) = \langle w, \phi(x) \rangle_{\mathcal{H}} \in \mathcal{Y}, \quad w \in \mathcal{H}$
- $\mathcal{F} = \{f_w | w \in \mathcal{H}\}$ RKHS with kernel $K\colon \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{Y}$, $K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$

Vector-valued case [1] $y_i = F(x_i) + \varepsilon_i, \quad F\colon \mathcal{X} \to \mathcal{Y}, \quad \mathcal{X}, \mathcal{Y}$ separable Hilbert spaces

- $\mathcal{H}$ parameter (Hilbert) space
- Feature map $\Phi\colon \mathcal{X} \to \mathcal{L}(\mathcal{H}, \mathcal{Y})$, search for $F_w(x) = \Phi(x)w \in \mathcal{Y}, \quad w \in \mathcal{H}$
- $\mathcal{F} = \{F_w | w \in \mathcal{H}\}$ RKHS with kernel $\mathcal{K}\colon \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{L}(\mathcal{Y}, \mathcal{Y})$, $\mathcal{K}(x, x') = \Phi(x)\Phi^*(x')$

Note that $\|f\|_{\mathcal{F}} = \|w\|_{\mathcal{H}}$

[1] See Houman Owhadi

Operator valued kernels for convolution

Vectorial case for convolution $y_i = \overbrace{F(x_i)}^{x_i * w} + \varepsilon_i, \quad F\colon \mathcal{X} \to \mathcal{Y}, \quad \mathcal{X} = L^1, \quad \mathcal{Y} = L^2$

- $\mathcal{H}$ (scalar) RKHS with $K(t,s) = K_0(t-s), \quad K \in L^1$
- Feature map $\Phi\colon \mathcal{X} \to \mathcal{L}(\mathcal{H}, \mathcal{Y})$, search for $F_w(x) = \Phi(x)w = x * w \in \mathcal{Y}, \quad w \in \mathcal{H}$
- $\mathcal{F} = \{F_w | w \in \mathcal{H}\}$ RKHS with kernel $\mathcal{K}\colon \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{L}(\mathcal{Y}, \mathcal{Y}), \quad \mathcal{K}(x, x') = C_{x * K_0 * \bar{x}'}$

Operator valued kernels for convolution

Vectorial case for convolution $y_i = \overbrace{F(x_i)}^{x_i * w} + \varepsilon_i, \quad F\colon \mathcal{X} \to \mathcal{Y}, \quad \mathcal{X} = L^1, \quad \mathcal{Y} = L^2$

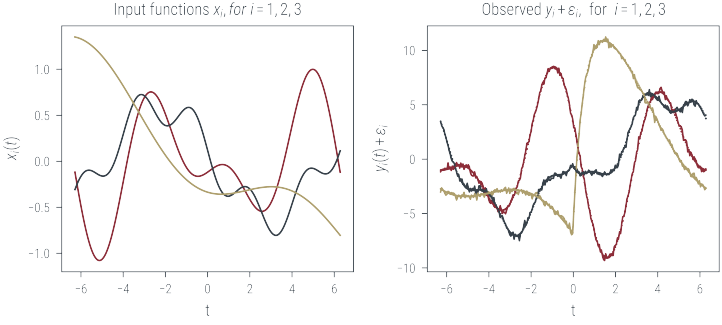
- $\mathcal{H}$ (scalar) RKHS with $K(t,s) = K_0(t-s), \quad K \in L^1$
- Feature map $\Phi\colon \mathcal{X} \to \mathcal{L}(\mathcal{H}, \mathcal{Y})$, search for $F_w(x) = \Phi(x)w = x * w \in \mathcal{Y}, \quad w \in \mathcal{H}$
- $\mathcal{F} = \{F_w | w \in \mathcal{H}\}$ RKHS with kernel $\mathcal{K}\colon \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{L}(\mathcal{Y}, \mathcal{Y}), \ \mathcal{K}(x,x') = C_{x * K_0 * \bar{x}'}$

Parametrizing in frequency domain makes computation easier!

- $\Phi\colon \mathcal{X} \to \mathcal{L}(\widehat{\mathcal{H}}, \mathcal{Y})$, with $\Phi(x)w = x * \mathcal{F}^{-1}\widehat{w}$
- $\mathcal{F}\mathcal{K}(x,x')\mathcal{F}^{-1}\colon l^2 \longrightarrow l^2$ is Fourier multiplier with $\left(\mathcal{F}\mathcal{K}(x,x')\mathcal{F}^{-1}\right)_{l\,l'} = \delta_{l\,l'}\widehat{x}\widehat{K_0}\widehat{x'}$

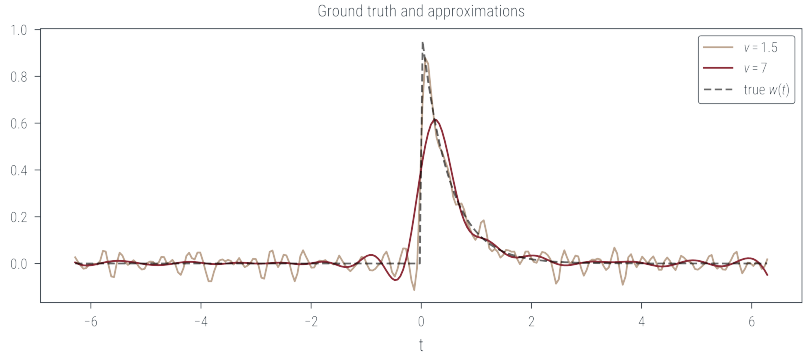
$$(\tilde{w})_l = [\ldots] = \frac{\frac{1}{n}\sum_{i=1}^n (\widehat{x}_i)_l (\widehat{y}_i)_l}{\frac{1}{n}\sum_{i=1}^n |(\widehat{x}_i)_l|^2 + \lambda \widehat{k}_l^{-1}}.$$

An example


Input functions x_i, for $i = 1, 2, 3$
$x_i(t)$
1.0
0.5
0.0
−0.5
−1.0
−6
−4
−2
0
2
4
6
t
Observed $y_i + \varepsilon_i$, for $i = 1, 2, 3$
$y_i(t) + \varepsilon_i$
10
5
0
−5
−10
−6
−4
−2
0
2
4
6
t

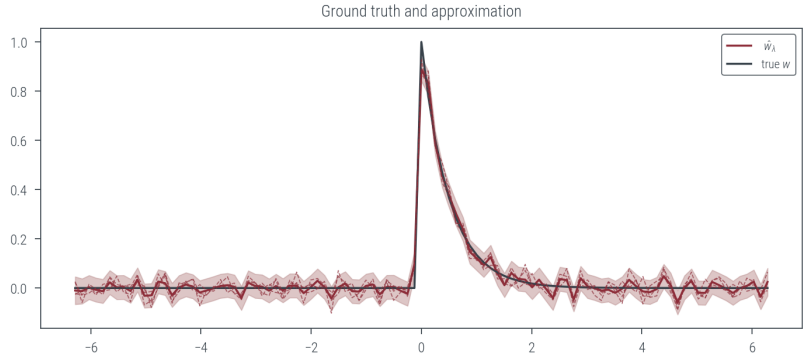
Changing kernel

i.e. changing RKHS/ GP covariance matrix



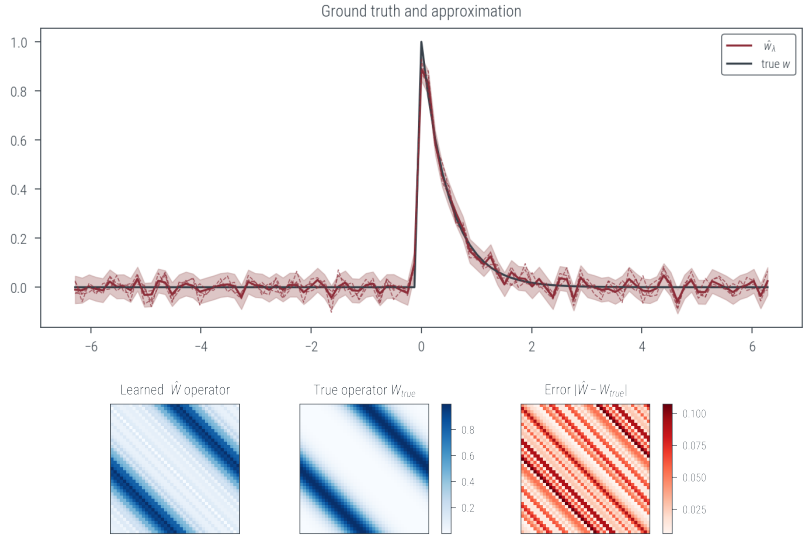
Changing kernel

i.e. changing RKHS/ GP covariance matrix



Changing kernel

i.e. changing RKHS/ GP covariance matrix



Source conditions for convolution

Convergence rates usually rely on source conditions. In general, it is difficult to formulate source conditions that are verifiable. This is not the case for convolution.

Theorem Assume $w^* \in L^2(G)$, let $(x_i, y_i)_{i=1}^N \overset{i.i.d}{\sim} \rho$ probability distribution on $\mathcal{X} \times \mathcal{Y}$,

$$\sum_{l \in \mathbb{Z}} \frac{|\widehat{w}_l^*|}{(\widehat{k}_l \mathbb{E}[|\widehat{X}_l|^2])^\alpha} < +\infty$$

and let

$$\tilde{w} = \arg\min_{w \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \|x_i * w - y_i\|_{\mathcal{Y}}^2 + \lambda_n \|w\|_{\mathcal{H}}^2$$

with regularization parameter $\lambda_n = n^{-\frac{1}{\alpha+2}}$. Then with probability at least $1 - 4e^{\tau}$

$$\|\tilde{w} - w^*\|_{L^2} \leq C\tau n^{-\frac{\alpha}{\alpha+2}}$$

Summary

the 'fil rouge'

- Solution operators for parametric PDEs
- GPs for inferring the Geen's function (inverse problems)
- Generalization to deep nets: Laplace approximation
- Operator learning from data: convolution operators

Summary

the 'fil rouge'

- Solution operators for parametric PDEs
- GPs for inferring the Geen's function (inverse problems)
- Generalization to deep nets: Laplace approximation
- Operator learning from data: convolution operators

Thank you!