Consider solving

$$\frac{\partial u}{\partial t} = F(u, Du, x, t)$$

$$u(x,0) = u_0(x)$$

ex: $u_0 = u_0(x) \in L^1(0,1)$

$u(0) = u(1) = 0$

guess

$$u(x,t) = \sum_{n=0}^{\infty} c_n \sin n\pi x$$

truncate at N & minimize

$$\min_{\vec{c} \in \mathbb{R}^N} \| u - u_N \|_2^2$$

solution:

$c_n = 2 \langle u_0, \sin n\pi x \rangle_{L^2(0,1)}$

This was a balker parametrization!

Here's a different approach...

use a neural network!

......

Let $r_\theta = \frac{\partial u_\theta}{\partial t} - \frac{\partial^2 u_\theta}{\partial x^2}$

$$L_1 = \frac{1}{N_1} \sum_{i=1}^{N_1} |r_\theta(x_i, t_i)|^2$$

$$L_2 = \frac{1}{N_2} \sum_{i=1}^{N_2} |u_\theta(x_i, 0) - u_0(x_i)|^2$$

$$L_3 = \frac{1}{N_3} \sum_{i=1}^{N_3} |u_\theta(0, t_i)|^2 + |u_\theta(1, t_i)|^2$$

- -

$m = \sum L_i$

$$\min_{\theta \in \mathbb{R}^m} \sum_{i=1}^{3} \alpha_i L_i$$

How we learn!

(*) So we learn the function

(*) But if we change $u_0(x)$?

(*) The solution map is way more useful!

ex: $u_0(x) \in L^1(\mathbb{R})$

Then

$$u(x,t) = (G_t * u_0)(x)$$

$$= \int_{\mathbb{R}} G(x-y, t)\, u_0(y)\, dy$$

$$G_t = \frac{1}{\sqrt{4\pi t}}\, e^{-x^2/4t}$$

What about forcing?

$$u_t = u_{xx} + f$$

$$u = G_t * u_0 + \int_0^t \int_{\mathbb{R}} G(x-y, t-s)\, f(y,s)\, dy\, ds$$

So much more useful than just one function!

(*) This shift from functions to operators is the key idea here

(*) So how do we approach learning an operator?

- Take inspiration from traditional deep learning
- Consider an NN layer

$$v_i^{l+1} = \sigma\left(\sum_{j=1}^{n} W_{ij} v_j^{l} + b_i\right)$$

activation

weights

bias

ReLu

What if we take $n \to \infty$ for $v^l = \langle v_1^l, v_2^l, \dots, v_n^l \rangle$?

$$\sum_{j=1}^{n} W_{ij} v_j^l \longrightarrow \int k(x_i, y) v^l(y)\, dy$$

(*) Dense layers are discretized integral operators

(*) A neural operator in this setting is just a NN with layer

$$v^{l+1}(x) = \sigma\left(A_\theta v^l(x) + \int_{\Omega} \kappa_\theta(x,y) v^l(y)\, dy\right)$$

(by analogy)

(*) But now that we are on function spaces... we can discretize back!

$$v^{l+1}(x_i) = \sigma\Big(A_\theta v^l(x_i) + \sum_{j=1}^{n} \kappa_\theta(x_i, x_j)\, v^l(x_j)\, w_j\Big)$$

quadrature weights

(*) If the PDE grid is irregular use a graph!

- Nodes = points x_i
- Edges connect x_i to nearby x_j

(*) $\eta_\theta(x_i, x_j)$ plays the role of an edge weight in this local "message-passing" scheme

(*) If we have a regular grid, then we can do better.
Especially with translation invariance

$$v^{l+1} = \sigma\left(A_\theta v^l + \eta_\theta * v^l\right)$$

you know what's coming

$$\widehat{\eta_\theta * v^l} = \hat{\eta}_\theta \hat{v}^l$$

Building the Fourier piece has 3 steps:

(i) $\hat{v}^l(k) = \mathcal{F}\{v^l\}(k)$

(ii) $\hat{v}^{l+1}(k) = R_\theta(k)\,\hat{v}^l(k)$

(*) and this is what you learn

(*) sometimes diagonal, sometimes a low pass filter...

(*) signal processing people valuable here

(iii) $\eta_\theta * v^l$

$= \mathcal{F}^{-1}\{\hat{\eta}_\theta \hat{v}^l\}(x)$

$= \mathcal{F}^{-1}\{R_\theta \hat{v}^l\}(x)$

$$v^{l+1} = \sigma\left(W_\theta v^l + \mathcal{F}^{-1}\{R_\theta \hat{v}^l\}\right)$$

(*) This is just one layer

(*) What's the whole structure?

Lift $\quad \mathbb{R}^{d_{in}} \rightarrow \mathbb{R}^{d}$ (Fourier) $\xrightarrow{\text{Project}}$

$v^0(x) = P_{lift}(a(x))$ input function

$v^{l+1} = \ldots$ (insert here)

$l = 0, 1, \ldots, L-1$

$u(x) = P_{proj}(v^L(x))$
(output decoding)

$G_\theta(a) = P_{proj} \circ$
$(\lambda_\theta^{(L)} \circ \ldots \circ \lambda_\theta^{(1)}) \circ P_{lift}(a)$

(!) This, the paper

claims, is the <u>only</u> resolution invariant neural operator that universally approximates

(*) The paper trashes DeepONets (branch/trunk)

Let's pose the learning framework...

- Let A and $\mathcal{U}$ be Banach spaces of functions on $D \subset \mathbb{R}^d$, $D' \subset \mathbb{R}^{d'}$

- Denote the true map as

$$G^{+} : \mathcal{A} \to \mathcal{U}$$

- Define the Bochner norm

$$\|f\|^2_{L^2_\mu(\mathcal{A};\mathcal{U})} = \int_{\mathcal{A}} \|f(a)\|^2_{\mathcal{U}} \, d\mu(a)$$

$$= \mathbb{E}_{a \sim \mu} \left\{ \|f(a)\|^2_{\mathcal{U}} \right\}$$

Theoretical : $\min_{\theta \in \mathbb{R}^p} \|G^{+} - G_\theta\|^2_{L^2_\mu(\mathcal{A};\mathcal{U})}$

Practical : $\min_{\theta \in \mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} \|u^{(i)} - G_\theta(a^{(i)})\|^2_{\mathcal{U}}$

(*) Define the discretization error as

$$|J_{\text{theoretical}} - J_{\text{practical}}|$$

Assume

* $\sup_x |\int g - \sum g| \leq C h^q$
* $\eta_{\theta,h}$ Lipschitz in h
* $\| S_h G^{+}(a) - G_{\theta^*}(S_h a) \|^2_{h,\mu} \leq \varepsilon^2$

then

$$\| S_{\tilde{h}} G^{+}(a) - G_{\theta^*}(S_{\tilde{h}} a) \|^2_{h,\mu}$$

$$\leq \varepsilon + \tilde{C}(h^q + \tilde{h}^q)$$

approximation error

training & test quadrature error

Universal Approximation

$\exists$ L, d, θ

depth width parameters

s.t.

$$\sup_{a \in K} \| G^{\dagger}(a) - G_{\theta}(a) \|_{\mathcal{U}} < \varepsilon$$

where $K \subset \mathcal{A}$
compact

$G^{\dagger} : \mathcal{A} \to \mathcal{U}$

continuous (with respect to the norm topology on $\mathcal{A}$ & $\mathcal{U}$)