1. PHYSICS-INFORMED NEURAL NETWORK (PINN) ⟶ first way to apply NN to PDE's solver, train in unsupervision way

From the previous lessons, NN are just "magic operator" applies on number to compute classes.

↓

We're moving on unsupervision learning so we do NOT need snapshots to compute the solution

For more information ⟶ • 1998: Lagaris first implementation
• 2018: Karniadakis today best implementation

What is the idea beyond the PINN? Create a NN that is able to achieve $\hat{u}$ ⟶ INPUT to NN: point of domain
OUTPUT of NN: evaluation of $\hat{u}$ in the inpur point

Our goal is to solve a PDE

(1) $\begin{cases} A(\hat{u}) = f & \text{on } \Omega \subset \mathbb{R}^d \text{ open, bounded, Lipschitz} \\ B(\hat{u}) = g & \text{on } \Gamma \subseteq \partial\Omega \end{cases}$ ⟶ boundary condition (strong)

with $f: \Omega \to \mathbb{R}$ and $g: \Gamma \to \mathbb{R}$ given

($\hat{u}$: solution)

NB • $A(\cdot)$ operator can represents stationary & evolutionary problems, moreover it can be non-linear, time-dependent
• $B(\cdot)$ is a boundary operator, i.e. trace operator or normal derivate

The idea of PINN is to build a NN that learns how to solve PDE

Hence, we want $\mathcal{J}_{NN}: \Omega \longrightarrow \mathcal{Y}$ with simulare FOM or using snapshots

$x \longmapsto u(x)$ approximation of $\hat{u}(x)$

$\underline{x} = \begin{pmatrix} x \\ y \\ t \\ \mu \end{pmatrix} \longmapsto u(x, y, t, \mu)$

it is a network with n inputs and d outputs

↓

We need physics to build the NN in an unsupervision way ⟶ we train $\mathcal{J}_{NN}$ unsupervised: we do NOT required snapshots

So, we create **loss function** using the residuals in the strong formulation

- $L_A^*(v, x) = \frac{1}{2} \left| f|_{\underline{x}} - A(v)|_{\underline{x}} \right|^2$
- $L_B^*(v, x) = \frac{1}{2} \left| g|_{\underline{x}} - B(v)|_{\underline{x}} \right|^2$

} Notice that we're NOT using weak formulation, but directly the strong formulation

and combined them via convex combination via λ Known a-priori

⟶ the **total loss function** is $\boxed{L^*(v, x) = L_A(v, x) + \lambda L_B(v, x)}$ that does not depends on snapshots

this is an hyperparameter (learn before the training) and balance the Order of magnitude of the error
you can find this by hand or automatically somehow

↓

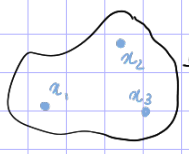
The solution is $\boxed{u_{NN} := \underset{v \in \mathcal{J}_{NN}}{\operatorname{argmin}} L^*(v, x)}$, hence we have to minimize this function ⟶ $u_{NN} \in \mathcal{J}_{NN}$

BUT it's NOT easy to find this solution in this way, due to non-linearity of equation or higher dimension (2D, 3D)

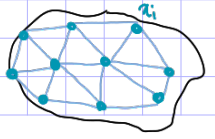
↳ The computation of L^* requires to strong hypothesis of f, g, A, B: one way to relax this is using integration
Thus we define a new loss function $L(v, x)$ given by $L(v) = L_A(v) + \lambda L_B(v)$ where $L_A(v) = \frac{1}{2}\int_\Omega |f - A(v)|^2$ & $L_B(v) = \frac{1}{2}\int_\Gamma |g - B(v)|^2$
and use a quadrature formula to approximate the integrals

How we can minimize the loss function? We have to compute some x to evaluate the loss function.
How to get quadrature points? There are different methods:

- **Meshless method** ⟶ similar to monte carlo method
We have the domain Ω, we randomly generate some $\{x_i\}$ and evaluate here the loss function



- **Mesh generation**
We discretize the space/domain and $\{x_i\}$ are the vertices of the mesh



1.1 Error PINN ⟶ what we can say about the error? The quality of u_{NN} depends on many factors

As we have something on the physics, we can measure the error that depends on 3 part:

1. Error capacity of NN : depends on the architecture of NN ⟶ number of layers, neurons

$$E_{NN} = \inf_{v \in \mathcal{Y}_{NN}} \|\hat{u} - v\|_X$$

⟶ best approximation of $\hat{u}$ in $\mathcal{Y}_{NN}$

2. Error associated to the loss function: associated to the minimization process

E_{Loss}

3. Error accuracy of the approximation of the residuals: ↗ related to the minimization problem

measure how good I am to compute L^*, so approximate correctly the physics ⇒ error integration

$$L^*(v) = \frac{1}{2}|f - A(v)|^2 + \frac{1}{2}|g - B(v)|^2$$

↓

Here we need strong hypothesis because we use strong formulation

BUT we can relate the loss function ↴

$$L^*_W(v) = \frac{1}{2|\Omega|}\int_\Omega |f - A(v)|^2 + \frac{1}{2|\Omega|}\int_\Gamma |g - B(v)|^2$$

weak formulation

⇒ disadvantages: we have to solve integral

↘ use this formulation in the project!

Hence, the final error of the PINN is

$$\|u_{NN} - \hat{u}\|_X \leq E_{NN} + E_{Loss} + E_{QUAD}$$

To relax more the hypothesis, we can study the

1.2 Variational PINN ⟶ can we do something more? Yes, to relax the hypothesis we can use the weak form of (1)

We're introducing the variational (weak) form in the loss function

$$L^*_{WW}(v) = \frac{1}{2|\Omega|}\int_\Omega f(w) - a(v, w) + \cdots \qquad \forall w \in V_R$$