Deep Learning for physics

or : incorporating priors (or physics) in a ML pipeline

ex : weather forecast ; dynamical system

1) Equations known

$$\frac{\partial x}{\partial t} = \underset{\text{coefficient to be estimated}}{\boxed{\nu}} \Delta x + \overset{\text{to be estimated}}{\boxed{v}} \cdot x$$

data assimilation
- statistical estimates
- Kalman filter

- or : equation known perfectly but chaotic system

2) Learn a PDE : Partial Differential Equation

- real equation not known
- genetic optimization (of possible terms to include)
- PDE : iterative process

$$\frac{\partial x}{\partial t} = f(x(t), \partial, \partial^2)(x))$$

$$\frac{\partial x}{\partial t} = f(x(t)) \longleftrightarrow \text{recurrent network}$$

$$\frac{x_{t+1} - x_t}{dt} \simeq f(x(t)) \underset{\text{approximation error}}{+ O(dt)}$$

$$x_{t+1} = x_t + dt\, f(x(t)) + O(dt^2)$$

$$x_{t+1} = f(x_t)$$

$$\Downarrow$$

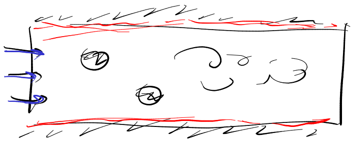
$$x_{t+1} = x_t + \varepsilon f(x_t)$$

3) Know an equation that the solution should satisfy

[DGM] Searching for a function $u : \underset{t}{[0,T]} \times \underset{x}{\Omega} \longrightarrow \mathbb{R}^n$

$$\begin{cases} \frac{\partial u}{\partial t}(t,x) = L u(t,x) & \forall t,x \in [0,T] \times \Omega \\ u(0,x) = u_0(x) \quad \forall x \in \Omega, & u(t,x) = g(t,x) \quad \forall x \in [0,T] \times \partial\Omega \end{cases}$$

Fluid mechanics :



$\Omega \subset \mathbb{R}^2$

ex : Navier-Stokes

Represent u by a neural network : $u : t, x \longmapsto u(t,x)$

$$u = F_\theta$$

"PINNs": physically-informed NN

at every location (x,t): $\left\| \frac{\partial u}{\partial t}(t,x) - Lu(t,x) \right\|$

$$\text{Loss}: \sum_{\text{samples }(x,t)} \left\| \frac{\partial u}{\partial t}(t,x) - Lu(t,x)\right\|^2 + \lambda \sum_{\substack{\text{samples}\\ x}} \|u(0,x) - u_0(x)\|^2$$

$$+ \lambda' \sum_{\substack{\text{samples}\\ (t,x)\in[0,T]\times\partial\Omega}} \|u(t,x) - g(t,x)\|^2$$

ex: chemistry: quantum mechanics
Schrödinger

4) Form of the solution known

Searching for (I_t) knowing $I_{t=0}$

$$" \quad \partial_t I + \underbrace{\overset{\text{unknown motion field}}{\vec{w}}\cdot\partial_x I}_{\text{advection (transport)}} = \underset{\text{real coefficient}}{D}\, \underbrace{\partial_{xx} I}_{\substack{\Delta I\\ \text{diffusion}}}$$

↳ Theorem: solution of the form:

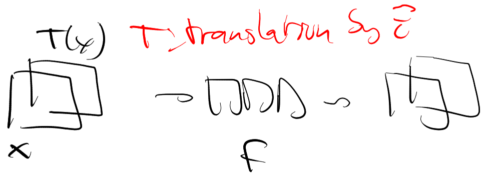
$$I(x,t) = \int_{\mathbb{R}^2} \underbrace{\frac{1}{4\pi D t} e^{-\frac{\|x-y-\vec{w}\|^2}{4Dt}}}_{\text{conv kernel}} I_0(y)\,dy \qquad k \qquad I_t = k * I_0$$

=> learn w
and predict $I = F(w)$
$\hookleftarrow$ fixed function above

↳ incorporating math knowledge in the ML pipeline

5) Incorporation of invariances / equivariances / priors by design

a) Translation-equivariance: CNN



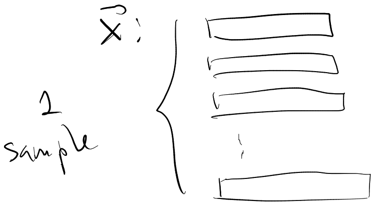
equivariance $F(T(x)) = T(F(x))$

invariance: $F(T(x)) = F(x)$

b) Rotation:

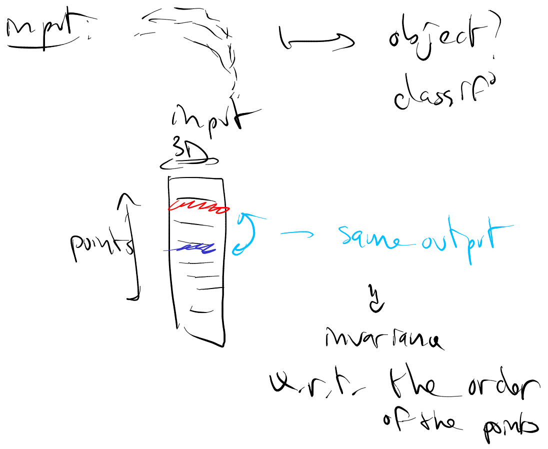
$$\begin{tikzcd} & \|\vec{v}\| \text{ norm} \arrow[r] & \varphi(\|\vec{v}\|) \arrow[dr] & \\ \vec{v} \arrow[ur] \arrow[dr] & & & \varphi(\|\vec{v}\|)\dfrac{\vec{v}}{\|\vec{v}\|} \quad \text{equivariant} \\ & \dfrac{\vec{v}}{\|\vec{v}\|} \text{ direction} \arrow[urr] & & \end{tikzcd}$$

c) Permutation-equivariance

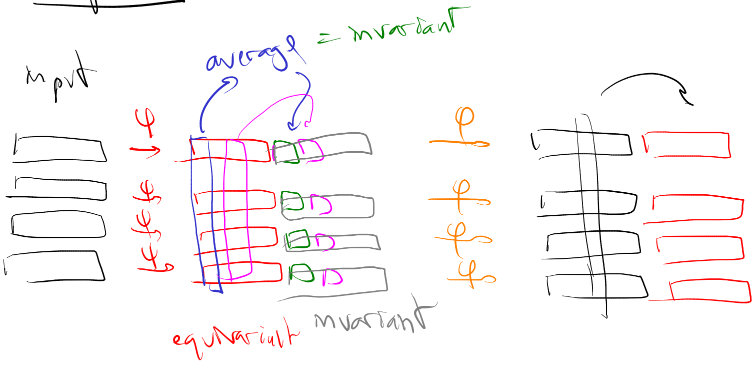


input: set of points
↳ order does not matter

eg: 3D point cloud



Deep Sets



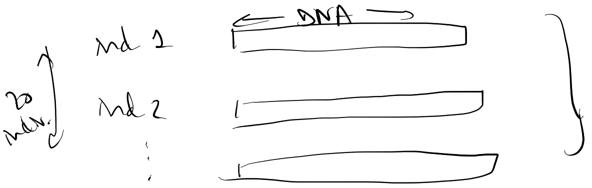
Guarantee by design

that the function is permutation-equivariant

Theorem: universal approximation power (to permutation-equivariant functions) provided many enough blocks

eg: population genetics

1 input = 1 population
= many individuals



→ predict: history of pop° size

eg: how many inhabitants 2000 years ago?

also: attention mechanism

d) Other properties enforced by design

eg: classif° task: desired property: output = probability distribution over N classes

p_c

$\forall c, \quad p_c \geq 0$

$p_c \leq 1$

$\sum p_c = 1$

→ achieved with softmax