Incorporating physical knowledge & modeling

Curriculum learning

- start an easier version of the task ⇝ hard

[Bengio 2009]

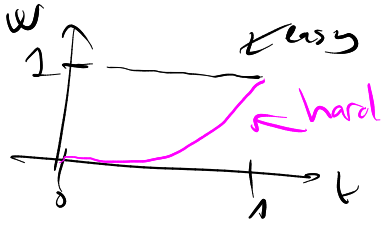
Training time

$t \in [0,1]$

same loss

but

distrib° getting more complex with time



$$q_t(z) = w_t(z)\, p(z)$$

q_t: sample

w_t: $0 \to 1$

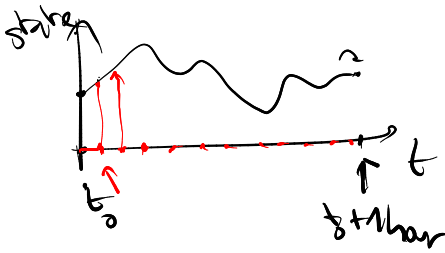
p: true distrib (of Dataset)

$t=1$

$w_{t=1}(z) = 1 \quad \forall z$

Distrib° entropy: $H(q_t)$ ↗ t

Ex of dynamical systems:



t ← time of the trajectory (predicted)

$$\text{Loss} = \sum_t \text{errors}$$

Invariances / Equivariance

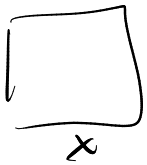
x = image

⟶ classificat° ↗ car ↘ dog

group of transforms = translations τ

$$\forall x \quad \forall \tau \qquad f(\tau(x)) = f(x)$$

invariance



x

⟶ segm° ⟶

$f(x)$

one label per pixel

$$f(\tau(x)) = \tau(f(x))$$

equivariance

How to?

- data augmentation: $x \to y$

$\tau(x) \to \tau(y)$ equiv NN

y

ML pipeline

demo

pros: easy to implement
- optim°

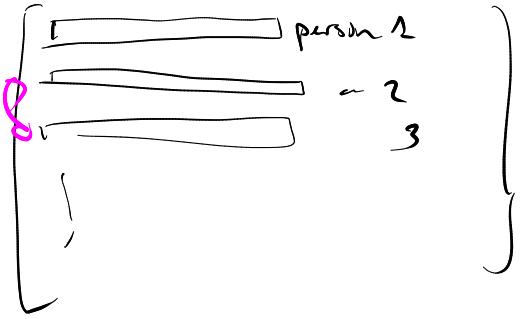
cons: no guarantee → approximate

- enforce exact invariance/equivariance by design

ex: translat° → CNN ∀ parameters

- permutat° - invar/equivar.

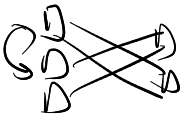
ex: DNA



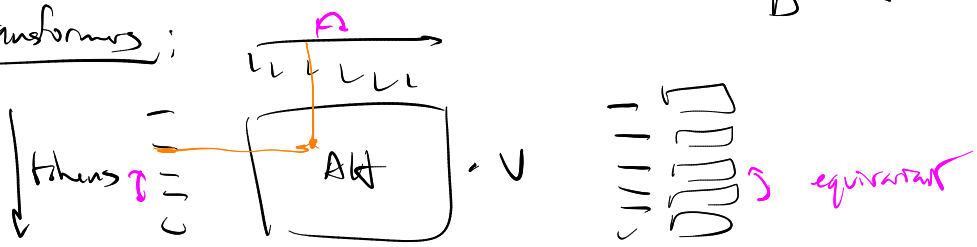
→ predict pop° genetics

→ size of pop° 10 000 years ago

1 input set



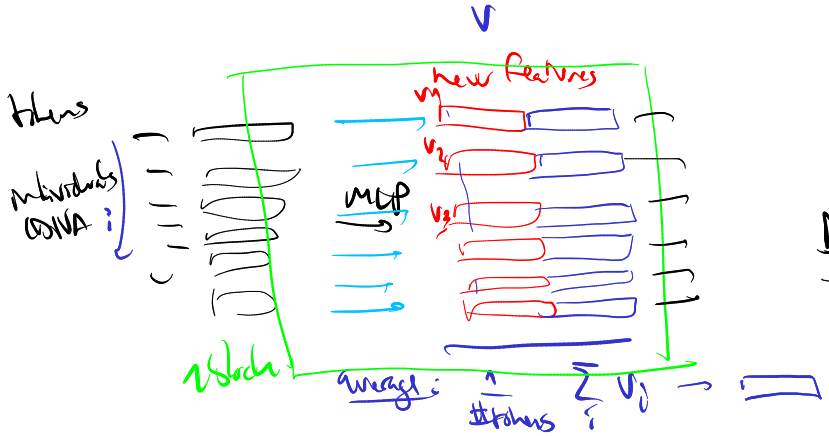
transformers:



MLP

applied mlp. to each token

making more complex features (independent for each token)



DeepSets [2017]

Universal approx th 1: $\forall$ permutat° invariant f, $\exists \phi, \psi$ tq $\forall x$, $f(x) = \psi\left(\sum_i \phi(x_i)\right)$ (average, outputs)

Univ approx th 2: by stacking many such blocks enough, we can express any permutation inv f.

Case of sets of 3D points:

list of 3D positions

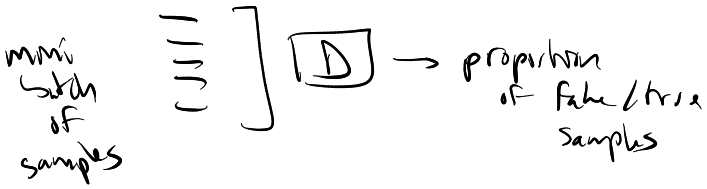
3D scan

point net (++)

set $\hookrightarrow$ graph ? mesh by joining nearest neighbors $\downarrow$ GNN

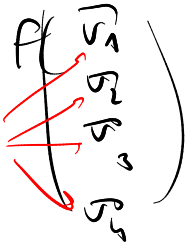
Improving GANS

usually:



Invariance/equiv to rotations

- easy way: $f(\vec{v})$ ↑ vector in 3D



$\vec{v}_1$ ↗

$\vec{v}_1 = \|v_2\|_x \frac{\vec{v}_2}{\|v_2\|}$

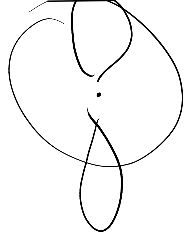
norm

direction

MLP

$g(\|v_2\|)\frac{\vec{v}_1}{\|v_2\|}$ equivariant to rotations

- hard way: theoretical physics: spherical harmonics



Bases of 3D fn on the sphere

Fourier

Clebsch-Gordan coeff

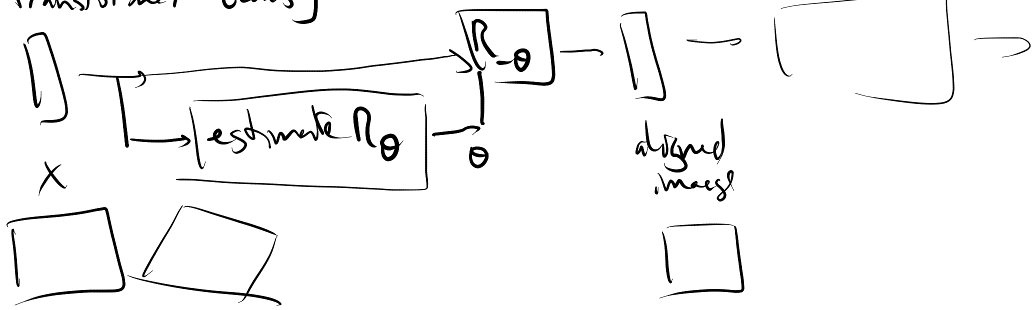
⇒ rotation-equiv

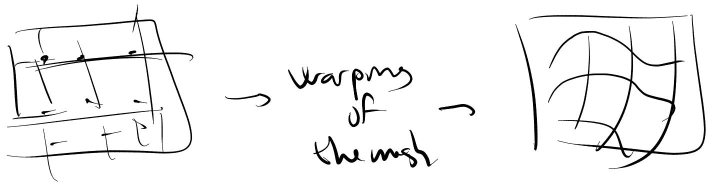
& full expressive power

(!) – rigid structure
– # of coeff to compute (truncate)

Learning the invariance

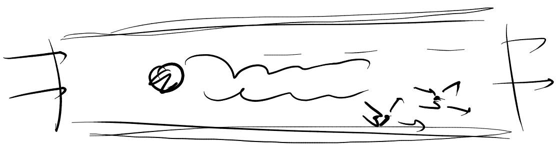
Spatial Transformer [2015]





Other property to enforce:

Fluid mechanics



incompressible

$\mathrm{div}\, f = 0$

$\mathrm{div}\, f(x) = 0 \quad \forall x$ point

$\mathrm{div} f = \frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y} \in \mathbb{R}$

$\binom{x}{y} \to$ NN $\to$ flow $f\binom{x}{y} = \binom{f_x}{f_y} \in \mathbb{R}^2$

location $\mathbb{R}^2$

div rot . = 0

$\nabla \cdot \nabla \wedge \, . = 0$

$\binom{x}{y} \to$ NN $\to z \binom{z_1}{z_2} \xrightarrow{\nabla \wedge} f(\vec{x})$

$g(\vec{x})$

$\left(\frac{\partial f_x}{\partial y}, -\frac{\partial f_y}{\partial x} \right)$

backprop

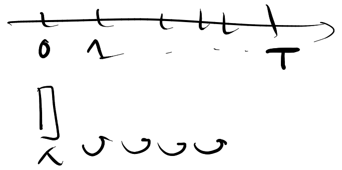
Dynamical systems

$$\frac{\partial u}{\partial t} = \alpha \Delta_x u + \mathrm{curl}(--)$$

↑ ? (under α) — viscosity GR

data assimilation
- eg: Kalman filter

Learning a PDE (Partial Differential Eq:)



x_0

$x_1 = f(x_0)$

$x_2 = f(x_1)$

⋮

, $x_{t+1} = f(x_t)$ — learn a network f

iterative process

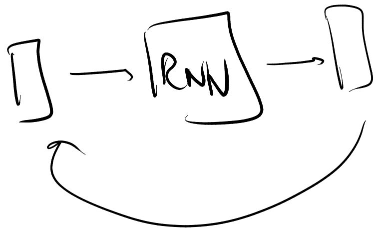
- expect $x_{t+2} \simeq x_t$

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

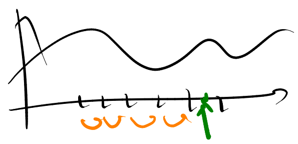
$f = Id + g$

- expect g does not depend on t

⇓ RNN (weights are shared across denoising steps)



More general case:



state

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

Hyp: constant time step — unit of t? time step

$$\rightarrow \quad x_{t+dt} = x_t + f(x_t)$$

↑ fixed time step — better unit (seconds)

$$\frac{x_{t+dt} - x_t}{dt} = f(x_t) \quad \longleftarrow \quad x_{t+dt} = x_t + dt\, f(x_t)$$

varying time step

$\lim_{dt \to 0} (--)$

predicting the derivative in the continuous limit

Δ accuracy of modeling

$$\frac{\partial x_t}{\partial t} = f(x_t) \longrightarrow \text{Taylor expansion: } x_{t+dt} = x_t + dt\, f(x_t) + \frac{1}{2} dt^2 H(x_t)$$

$\longleftarrow + O(dt^2)$

↳ numerical simulations "solvers": integrate

$$x_T = \int_{t=t_0}^{T} f(x_t)\, dt + x_{t_0}$$

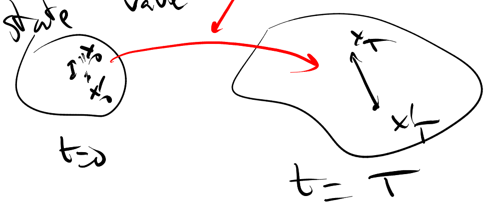
Runge-Kutta

Classical "integrators": automatically adapt the time discretiz° of the integral to the speed / changes

⚠ chaotic

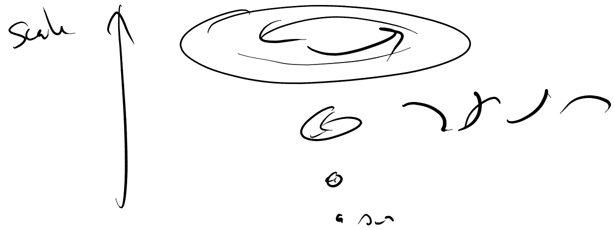
Lyapunov exponent

max eigen value of J $= |||J|||$



$$\max_{x_0, x_0'} \frac{\|x_T - x_T'\|}{\|x_0 - x_0'\|}$$

Turbulence: multi-scale



Deep learning:

$$\frac{\partial x}{\partial t} = f_\theta(x)$$ ← learn parameters

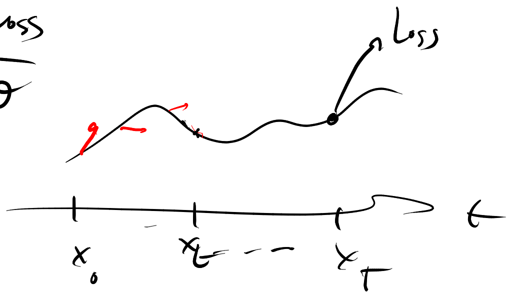
s.t. state $\hat{x}_T \simeq x_T$ real one

$\int_{t_0}^{T} f_\theta(x)\,dx$; $\int_{t_0}^{T} f(x)\,dy$ true eqn

$$\text{Loss}: \mathbb{E}_{\mathcal{D}} \left\| \int_{t_0}^{T} f_\theta(x)\,dx + x_{t_0} - x_T \right\|^2$$

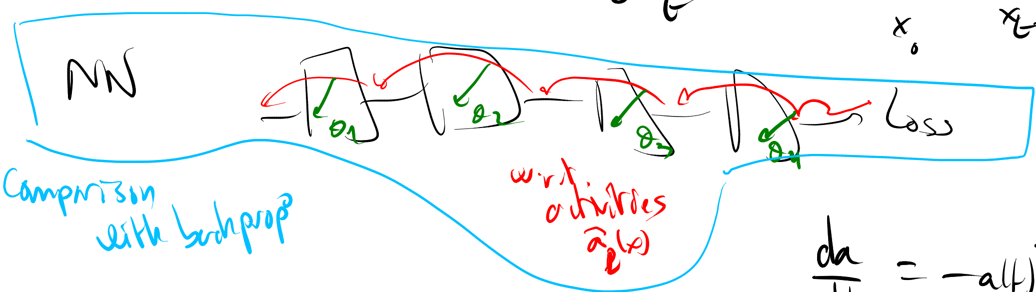
(Integrator (f_θ))

$\rightarrow \frac{\partial \text{Loss}}{\partial \theta}$



Adjoint method

$$a(t) = \frac{\partial L}{\partial x_t}$$



Comparison with backprop

$$\frac{da}{dt} = -a(t)^T \frac{\partial f_\theta(x_t)}{\partial x_t}$$

Neural ODE = NODE

$$\frac{dL}{d\theta} = -\int_{T}^{t_0} a(t)^T \frac{\partial f_\theta(x_t)}{\partial \theta}\,dt$$

↳ call the integrator to estimate this

In practice:
go to latent repr° first
(do that inside an autoencoder)

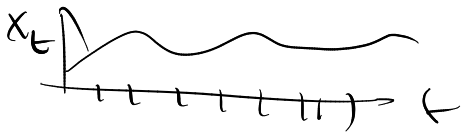
Koopman – von Neumann

dynamical system : $x_{t+1} = F(x_t)$

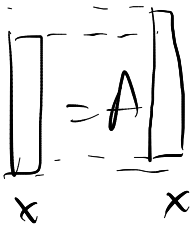
dim = $+\infty$

Th : $\forall$ dynamical system, $\exists$ representation space
s.t. the system becomes linear.

- F is linear : matrix : $x_{t+1} = A x_t \quad x_t = e^{tA} x_0$ if A (t)



$x_1 = A x_0$
$x_2 = A x_1$
$x_3 = A x_1$
!



↙ solve mean square pb

$\inf_A \sum_t \| x_{t+1} - A x_t \|^2$ ← $\inf_A \| X_{+1} - AX \|^2$

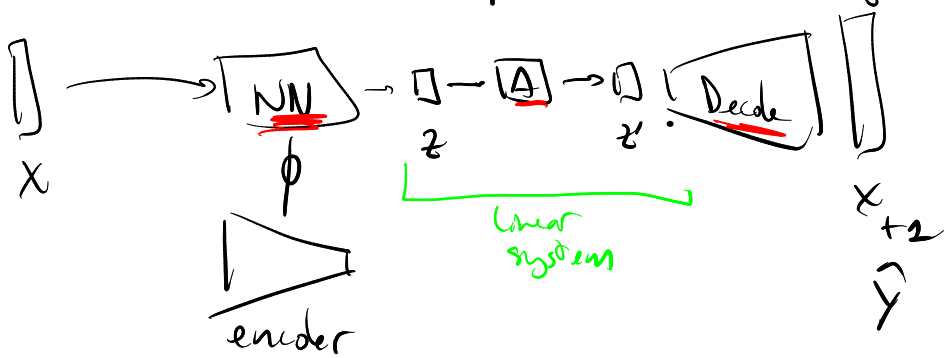
↓ easy

↳ SVD : pseudo-inverse +

optimal $A = X_{+1} X^+$ ← SVD QR

- Koopman – von Neumann :

if F non-linear : $\exists \phi$: $z = \phi(x)$ follows linear dynamics

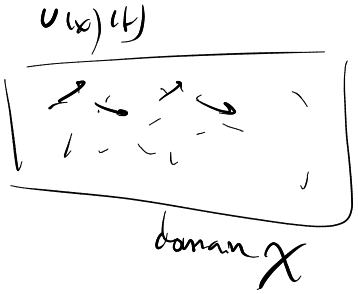


Loss : $\| X - \text{Dec} \circ \text{Enc}(X) \|^2$ auto-encoder quality

$+ \| y - \text{Dec} \circ A \circ \text{Enc}(X) \|^2$

NB : best A $|_{\text{given Enc Dec}}$ is obtained closed-form

PINNs : Physically-Informed NN



Hyp : know the eq° train NN to imitate it

eg :
$$\begin{cases} \partial_t u(t,x) + \mathcal{L} u(t,x) = 0 \quad \forall t,x \\ \text{boundary constraints :} \\ u(t=0,x) = u_0(x) \text{ known} \quad \forall x, t=0 \\ u(t, x \in \partial\Omega) = g(t,x) \text{ known} \quad \forall t \end{cases}$$

↳ operator : linear or not
↳ Green fct

heat eq° :
$$x_T = N(0,t) * x_0$$
kernel

without any training data

$$\text{Loss} = \| \partial_t u(t,x) + \mathcal{L} u(t,x) \|^2 \quad \forall x,t$$
$$+ \| u(0,x) - u_0(x) \|^2$$
$$+ \| u(t,x) - g(t,x) \|^2 \quad \text{for } x$$

Deep Galerkin Model

$$x, t \longrightarrow \boxed{NN_\theta} \longrightarrow u(t,x)$$

=> compute $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial t}$ ... backprop

- Neural Operator : learn the Green fct
  - GNN → kernel
    - Transformer

DeepONet