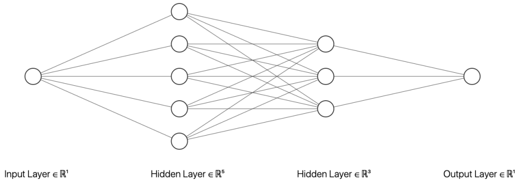
Differentiability Properties of Non-smooth Functions Used in Neural Networks

B. Després (LJLL-SU+MEGAVOLT-INRIA)

Take a fully-connected **feed-forward neural network** function $f : \mathbb{R}^{a_0} \to \mathbb{R}^{a_{\ell+1}}$

$$f = f_\ell \circ S_\ell \circ f_{\ell-1} \circ S_{\ell-1} \circ \cdots \circ f_1 \circ S_1 \circ f_0$$

where $f_i(x_i) = W_i x_i + b_i \in \mathbb{R}^{a_{i+1}}$ $(x_i \in \mathbb{R}^{a_i})$ is linear and $S_i \in C^0(\mathbb{R})$ is a non linear activation function $0 \leq S_i'(x) \leq 1$.



- Typically ReLU function $S_i(x) = R(x) = \max(0, x)$ is an extremely popular activation function used in production codes.

- *Max-pooling* used in **convolutional neural networks** is based on the maximum function $(a, b, \dots) \mapsto \max(a, b, \dots) \in \mathbb{R}$.

Autodiff

We need to differentiate f :

- with respect to x : for the assembly of the cost function for approximation of PDEs (DeepRitz, PINNS, VPINNS, ..).
- with respect to the weights (W_i, b_i) : for training.
  This is performed with SGD and autodiff.
- with respect to the weights (W_i, b_i) inside a numerical hybrid model.
  For example NeuralGCM and future hybrid transport models.

Principle

All derivatives are exactly calculated with
automatic differentiation=chain rule=backprop
which is the main tool for **composition of functions**.

This is the key technic in Tensorflow, Pytorch, Jax, ScikiLearn, ...

The standard chain rule hold for C^1 functions :
take $J = v \circ u$ where $v, u \in C^1$, then

$$\nabla J = \nabla v \circ u \; \nabla u \Longleftrightarrow \nabla J(W) = \nabla v(u(W)) \; \nabla u(W).$$

Regularity of NNs

- Clearly $f \in \mathrm{Lip}(\mathbb{R}^{a_0} : \mathbb{R}^{a_{\ell+1}})$.
The Rademacher theorem states $\nabla f \in L^\infty(\mathbb{R}^{a_0} : \mathcal{M}_{a_{\ell+1},a_0}(\mathbb{R}))$.

- ML is universally based on the chain rule

$$\nabla f(x) = W_\ell B_\ell(x) W_{\ell-1} B_{\ell-1}(x) \dots W_1 B_1(x) W_0$$

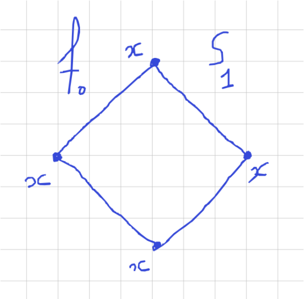
which is non ambiguous for C^1 activation functions.

- Everything is Lipschitz-continuous, one can write

$$B_r = \nabla S_r \circ \cdots \circ S_1 \circ f_0 \in L^\infty(\mathbb{R}^{a_0} : \mathcal{M}_{a_r,a_0}(\mathbb{R})).$$

$\Rightarrow$ It seems there is no problem.

Why there is a problem (Murat-Trombetti 2003)



It is an <u>*academic one-hidden-layer CNN*</u> : $f_0 : \mathbb{R} \to \mathbb{R}^2$ is linear

$$f_0(x) = (x, x) = W_0 x \text{ with } W_0 = (1, 1)$$

and S_1 is a max-pooling function over two values

$$S_1(y) = \max(y_1, y_2), \text{ where } y = (y_1, y_2) \in \mathbb{R}^2.$$

Of course $f(x) = x$ and $f'(x) = 1$.

The gradient of f_0 is $\nabla f_0 = W_0 = (1, 1)$.
The gradient of S_1 is defined almost everywhere

$$\begin{cases} \text{if } y_1 > y_2 & \nabla S_1(y) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbb{R}^2, \\ \text{if } y_1 < y_2 & \nabla S_1(y) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \in \mathbb{R}^2, \\ \text{if } y_1 = y_2 & \nabla S_1(y) \text{ is not defined (in } \mathbb{R}^2). \end{cases}$$

Therefore the chain rule becomes meaningless

$$1 = \langle \nabla S_1 \circ f_0(x), (1, 1) \rangle .$$

The key issue is :
the pre-image of
sets with zero measure might be
sets with non zero measure.

A fundamental result

F. Murat and C. Trombetti. A chain rule formula for the composition of a vector-valued function by a piecewise smooth function. 2003.

Consider a finite decomposition of $\mathbb{R}^a$ in Borel sets or Borel pieces P^α

$$\mathbb{R}^a = \bigcup_\alpha P^\alpha, \qquad P^\alpha \cap P^\beta = \varnothing \text{ for } \alpha \neq \beta.$$

For the sake of simplicity, replace Borel by piecewise affine.

Definition

A Lipschitz-continuous function $f : \mathbb{R}^a \to \mathbb{R}^b$ is

$$C^1_{\mathrm{pw}} = C^1 \text{ piecewise}$$

if there exists Borel pieces P^α and functions $f^\alpha \in \mathrm{Lip}(\mathbb{R}^a : \mathbb{R}^b) \cap C^1(\mathbb{R}^a : \mathbb{R}^b)$ such that

$$f(x) = \sum_\alpha \mathbf{1}_{P^\alpha}(x) f^\alpha(x) \quad \forall x \in \mathbb{R}^a.$$

Here $f^\alpha(x) = f(x)$ for all $x \in P^\alpha$ and $\nabla f^\alpha \in L^\infty(\mathbb{R}^a : \mathcal{M}_{b,a}(\mathbb{R})) \cap C^0(\mathbb{R}^a : \mathcal{M}_{b,a}(\mathbb{R}))$.

Associated gradients

Definition (Associated gradients)

We say that the gradient associated to the representation is

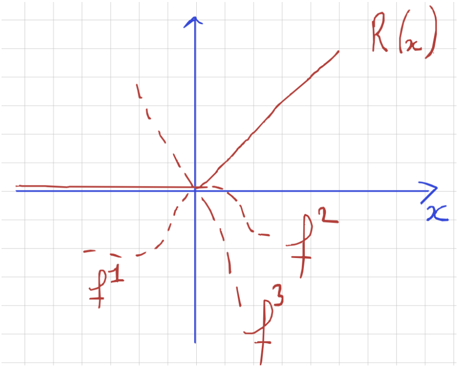
$$\widetilde{\nabla} f(x) = \sum_{\alpha} \mathbf{1}_{P^\alpha}(x) \nabla f^\alpha(x) \in \mathcal{M}_{b,a}(\mathbb{R}) \quad \forall x \in \mathbb{R}^a.$$

In brief it is an associated gradient.

The associated gradient is a real $b \times a$ matrix defined everywhere, that is for all $x \in \mathbb{R}^a$.
The associated gradient is non unique because it depends on the representation.

Extremely close to the notion of intentional derivative :
Lee-Yu-Rival-Yang, On Correctness of Automatic Differentiation for Non-Differentiable Functions, 2020.

ReLU$=R$



$P^1 =]0, \infty[$ and $f^1(x) = x$,
$P^2 =]-\infty, 0[$ and $f^2(x) = 0$,
$P^3 = \{0\}$ and f^3 is any C^1 function.
So $\widetilde{R}'(x) = 1$ for $x > 0$, $\widetilde{R}'(x) = 0$ for $x < 0$ and $\widetilde{R}'(0) =$ any $z \in \mathbb{R}$.

Murat-Trombetti Theorem ($u \in H^1$)

Theorem (D. TMLR 2025, in the context of ML)

Consider two functions. The first one $u \in \mathrm{Lip}(\mathbb{R}^a : \mathbb{R}^b)$ is Lipschitz-continuous. The second one $v \in \mathrm{Lip}(\mathbb{R}^b : \mathbb{R}^c)$ is Lipschitz-continuous and C^1_{pw} with a representation with an associated gradient.

Then the chain rule identity holds in $L^\infty(\mathbb{R}^a : \mathcal{M}_{c,a}(\mathbb{R}))$

$$\nabla(v \circ u) = \widetilde{\nabla} v \circ u \ \nabla u$$

where $\widetilde{\nabla} v \circ u(x) = \widetilde{\nabla} v(u(x))$ for all $x \in \mathbb{R}^a$.

- The proof is ultimately based on a Stampacchia formula.
- Borel pieces is the optimal regularity.
- An associated gradient is a representative of the gradient in the sense of distributions.

An algebra

Theorem (D. TMLR 2025)

Consider a Neural Network function f with all functions f_r and S_r Lipschitz-continuous for $1 \leq r \leq \ell$.
Assume that **all** f_r **and** S_r **are** C^1_{pw} with an associated gradient.
Then f itself has an associated gradient

$$\widetilde{\nabla} f = W_\ell \widetilde{B}_\ell(x) W_{\ell-1} \widetilde{B}_{\ell-1}(x) \ldots W_1 \widetilde{B}_1(x) \widetilde{W_0} \quad a.e. \;\; x \in \mathbb{R}^a$$

where the right hand side is defined for all $x \in \mathbb{R}^a$, and

$$\widetilde{B}_r = \widetilde{\nabla} S_r \circ \cdots \circ S_1 \circ f_0 \in \mathcal{M}_{a_r}(\mathbb{R}) \text{ for all } x \in \mathbb{R}^a.$$

The space of functions $\mathrm{Lip}(\mathbb{R}^a : \mathbb{R}^a) \bigcap C^1_{\mathrm{pw}}(\mathbb{R}^a : \mathbb{R}^a)$ is an **algebra.**

So far, this space does not come with a metric.

I : Boustany example (better say Boustany paradox)

Ryan Boustany. On the numerical reliability of nonsmooth autodiff : a maxpool case study. Transactions on Machine Learning Research, 2024. URL `https://arxiv.org/abs/2401.02736`

Bolte and Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. Mathematical Programming, 188 :19-51, 2021.

For $x = (1, 2, 3, 4)$ assemble in Pytorch the function

$$t \mapsto f(t) = \max_1(tx) - \max_2(tx)$$

```
def max₁(x):
res = x[0]
for i in range(1, a):
if x[i] > res:
   res = x[i]
return res
```

```
def max₂(x):
return torch.max(x)
```

TABLE – Script of the functions $\max_1$ and $\max_2$

Funny isn't it ?

Pytorch (TensorFlow, Scikilearn, ...,)

t	-1	-0.5	-0.01	0	0.01	0.5	1
$f(t)$	0	0	0	0	0	0	0
$f'(t)$	0	0	0	-1.5	0	0	0

TABLE – PyTorch-autodiff of $f(t)$

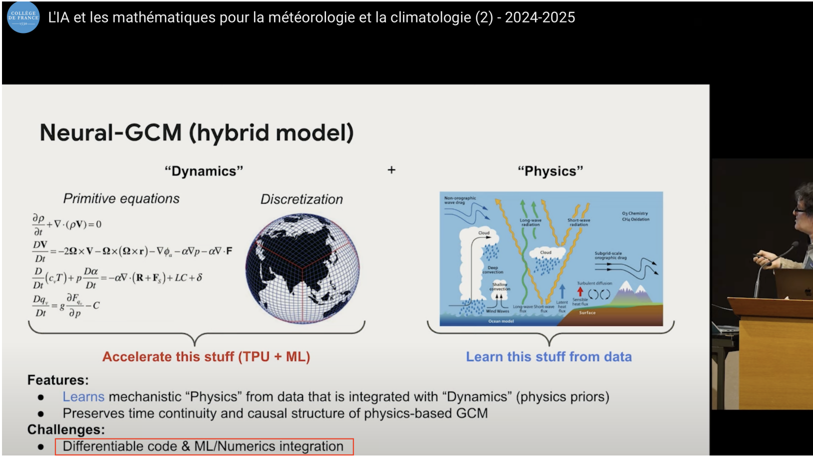
Solution of the paradox : representations and associated gradients are different

$$\widetilde{\nabla} \underset{1}{\max}(0,0,0,0) = (1,0,0,0) \text{ and } \widetilde{\nabla} \underset{2}{\max}(0,0,0,0) = \left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right)$$

Remark

Sub-differential theory and Clarke's generalized gradients do not help that much to describe this.

II : Hybrid Models in meteorology



Michael Brenner (Harvard+Google)

Neural general circulation models for weather and climate, Kochkov-et-al, Nature, 2024.

An attempt to formalize Hybrid Models

$$\begin{cases} \partial_t U + \mathcal{L}(U, \partial_x U, \partial_{xx} U, \dots) = S_W^{\mathrm{NN}}(\alpha), & W \text{ contains all NN } \textbf{weights}, \\ U(x, 0) = \beta, & \text{initial data at time } t = 0, \end{cases}$$

where one has a large **dataset of observations**

$$\mathcal{D} = \{(\alpha_i, \beta_i, \gamma_i) \mid i = 1, 2, \dots, N\} \subset \mathbb{R}^{m+n+n}.$$

The cost function evaluated at given final time $T > 0$ is something like

$$J(W) = \frac{1}{N} \sum_i \int_{x \in \Omega} |U(x, T : \alpha_i, \beta_i, W) - \gamma_i|^2 \, dx \qquad (+ \text{ many other terms}).$$

Principle

The sources $\alpha \mapsto S_W^{\mathrm{NN}}(\alpha)$ must be compatible with autodiff.
The discrete solution must also be compatible with autodiff.

Transport (essential for CFD)

- $t \geq 0,\ x \in \mathbb{R}: \quad \partial_t u + a \partial_x u = 0$. The upwind scheme is

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a \frac{u^n_{j+\frac{1}{2}} - u^n_{j-\frac{1}{2}}}{\Delta x} = 0,$$

where $u^n_{j+\frac{1}{2}} = G_a(u_j^n, u_{j+1}^n)$ follows the characteristic lines

$$G_a(\alpha, \beta) = \alpha \text{ if } a > 0,\ \ G_a(\alpha, \beta) = \beta \text{ if } a \leq 0.$$

The time steps are composed one after the other.

- What matters is the regularity of the flux

$$F(a, \alpha, \beta) = a G_a(\alpha, \beta) = R(a)\alpha - R(-a)\beta.$$

The function F is **(loc-)Lipschitz and** C^1_{pw}

Principle

ReLU and **non smooth activation functions**
are everywhere in CFD and transport models.

One more scenario : autodiff of high order schemes

Consider the 2nd order Lax-Wendroff scheme

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + a\frac{u^n_{j+\frac{1}{2}} - u^n_{j-\frac{1}{2}}}{\Delta x} = 0,$$

where the Courant number is $\nu = |a|\frac{\Delta t}{\Delta x}$ and

$$\begin{cases} a \geq 0 & u_{j+\frac{1}{2}} = u_j + \frac{1}{2}(1-\nu)\mathrm{minmod}(u_{j+1} - u_j, u_j - u_{j-1}), \\ a < 0 & u_{j+\frac{1}{2}} = u_{j+1} + \frac{1}{2}(1-\nu)\mathrm{minmod}(u_{j+1} - u_j, u_j - u_{j-1}). \end{cases}$$

One can check that

$$\mathrm{minmod}(x, y) = R(-\max(-x, -y)) - R(-\max(x, y)).$$

Lemma

The 2nd order Lax Wendroff scheme satisfies the autodiff requirements inherited from the C^1_{pw} framework.