SciML 2: Neural operators

E. Franck

Master CSMI, Strasbourg University

01/09/2025

1. Operator learning: principle

Contexte

- Les méthodes PINNs et leurs versions temporels sont des **méthodes numériques** qui permettent de résoudre des EDP dans une configurations ou une famille de configurations données.

Objectives (Modèles réduits ou simplifiés): On voudrait être capable de prédire une solution dans de grandes familles de cas tests. Deux stratégies:

- Reduction de modèles (cours de C. Prudhomme + 2 cours et un TP ici)
- Approximation d'opérateurs (2 cours + 2 TP)

- On se donne une EDP elliptique comme l'équation de Poisson:

$$\begin{cases} -\Delta u = f \quad \forall \mathbf{x} \in \Omega \\ \\ \alpha \langle \nabla u . \mathbf{n} \rangle + \beta u = g(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\Omega \end{cases}$$

avec $f \in L^2(\Omega)$.

Remark (Operator inverse): Puisqu'il existe une unique solution cela veut dire qu'il existe un opérateur continu $T^+ : L^2(\Omega) \to H^1(\Omega)$ tel que $T^+(f) = u$.

Objectifs

Objectives (Apprentissage d'opérateur): Apprendre T^+ pour un ensemble plus ou moins grand de source (ou autres données).

- **Plan**:
  - Exemple pour justifier l'architecture des réseaux approchant des opérateurs.
  - Architectures générales aux niveaux continu
  - Apprentissage d'opérateur
  - Principe de construction des architectures standards en dimension infinie
  - Introduction de leurs disctrétisation.

Definition (Noyau de Green dépendant du domaine pour le Laplacian): Le noyau de Green pour le Laplacian associé au domaine Ω est la fonction $G_\Omega : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ solution du problème

$$\begin{cases} -\Delta_{\mathbf{y}} G_\Omega(\mathbf{x},\mathbf{y}) = \delta(\mathbf{x}-\mathbf{y}) \quad \forall \mathbf{x},\mathbf{y} \in \Omega \\ \\ \alpha \langle \nabla G_\Omega(\mathbf{x},\mathbf{y}).\mathbf{n} \rangle + \beta G_\Omega(\mathbf{x},\mathbf{y}) = 0 \quad \forall \mathbf{y} \in \partial\Omega \end{cases}$$

Théorème de Green

Theorem (Opérateur de Green dépendant du domaine pour le Laplacian): Soit $f \in L^2(\Omega), g(\mathbf{x}) \in L^2(\partial\Omega)$, alors on a

$$u(\mathbf{x}) = T^+(f)(\mathbf{x})$$

est solution de l'EDP précédente. L'opérateur T^+ est donné par

$$T^+(f)(\mathbf{x}) = \int_\Omega G_\Omega(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y} - \int_{\partial\Omega} \langle \nabla_{\mathbf{y}} G_\Omega(\mathbf{x}, \mathbf{y}), \mathbf{n} \rangle g(\mathbf{y}) d\mathbf{y})$$

si $\alpha = 0$ and $\beta = 1$ et

$$T^+(f)(\mathbf{x}) = \int_\Omega G_\Omega(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y} + \int_{\partial\Omega} g(\mathbf{y}) G_\Omega(\mathbf{x}, \mathbf{y}) d\mathbf{y}$$

si $\alpha = 1$ and $\beta = 0$.

- Il existe une autre version (voir polycopié) on utilise le Noyau dans un domaine infini + une solution homogène.

Ingredient de base

- Pour construire l'opérateur inverse de notre l'ingrédient clé va être **l'opérateur à noyau**. Les opérateurs à noyau linéaire peuvent s'interpréter comme la généralisation en dimension infini des matrices.

Remark: Soit un opérateur:

$$u(x) = \int_{\Omega} G(x, y) f(y) dy$$

Si vous discrétiser, avec une méthode de quadrature un opérateur a noyau vous obtenu

$$u(x_i) = \sum_{i=1}^{q} w_i k(x_i, x_j) f(x_j)$$

qui se réécrit $u = Af$ avec $A_{ij} = w_i k(x_i, x_j)$

- On comprend bien que ses opérateurs seront la brique de base de réseaux de neurones en dimension infinie.

Couche générales

- On propose ici de définir de type de couches générales qui vont paramétrer des transformations de fonctions.

Definition (Nonlocal parametric operator): On se donne une fonction $\mathbf{v}_l : \mathbb{R}^d \to \mathbb{R}^{in}$ et $\mathbf{v}_{l+1} : \mathbb{R}^d \to \mathbb{R}^{out}$. On appelle une couche non-locale l'opérateur

$$\mathbf{v}_{l+1}(\mathbf{x}) = \mathcal{B}_{\boldsymbol{\theta}}(\mathbf{v})(\mathbf{x}) = \int_D k_\theta(\mathbf{x}, \mathbf{y}, \mathbf{v}_l(\mathbf{x}), \mathbf{v}_l(\mathbf{y}))\mathbf{v}_l(\mathbf{y})d\mathbf{y}$$

ou K_θ le noyau sera la partie apprenable. Dans la majorité des cas l'opérateur est linéaire pour généraliser les couches affines des réseaux classiques.

Definition (Local parametric operator): On se donne une fonction $\mathbf{v}_l : \mathbb{R}^d \to \mathbb{R}^{in}$ et $\mathbf{v}_{l+1} : \mathbb{R}^d \to \mathbb{R}^{out}$. On appelle une couche non-locale l'opérateur

$$\mathbf{v}_{l+1}(\mathbf{x}) = nn_{\boldsymbol{\theta}}(\mathbf{v}_l(\mathbf{x})), \forall \mathbf{x} \in \mathbb{R}^{in}$$

avec $nn_{\boldsymbol{\theta}}$ un réseau de neurones ou une couche paramétrée par $\boldsymbol{\theta}$.

Discrétisation

Definition (Réseau en dimension infini): Soit des espaces de fonctions H_x et H_y. Un réseau en dimension infini $\Phi : H_x \to H_y$ est une composition de couche locale et non locale.

Definition (Opérator paramétrique discret non local): On se donne $\mathbf{v} \in V \subset \left[L^2(\Omega)\right]^p$. On appelle $\mathcal{B}_\theta^h(\mathbf{v})$ un opérateur discret la transformation

$$\mathcal{B}(\mathbf{v})^h(\mathbf{x}) = \frac{1}{n} \sum_{i \in D_h} w_i k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y_i}, \mathbf{v}(\mathbf{x}), \mathbf{v}(\mathbf{y_i}))\mathbf{v}(\mathbf{y_i}).$$

Definition (discrétisation d'opérateur convergente): On se donne $\mathbf{v} \in V \subset \left[L^2(\Omega)\right]^p$. On se donne $\mathcal{B}_{\boldsymbol{\theta}}(\mathbf{v})$ et un opérateur discret $\mathcal{B}_\theta^h(\mathbf{v})$. On dit qu'un d'opérateur discret non local est convergent si

$$\| \mathcal{B}_\theta(\mathbf{x}) - \mathcal{B}_\theta^h(\mathbf{x}) \|_{L_\infty(\Omega)} \underbrace{\to}_{n \to \infty} 0, \quad \forall \mathbf{x} \in \Omega, \forall \mathbf{v} \in V$$

réseaux de neurones vs opérateurs neuronaux

Definition (Operateurs neuronals) : Un operateur neuronale sera une composition de discrétisation d'opérateurs paramétriques non locaux et locaux ou l'ensemble des donne une discrétisation convergente d'une composition d'opérateur.

En gros va différentier deux types d'architectures:

- les réseaux de neurones qui seront des discrétisation **non convergentes** d'opérateur continue et qui **travaillerons a resolution/discrétisation fixée**
- les opérateurs neuronaux qui seront des discrétisation **convergentes** d'opérateur continue et qui **pourront travailler a resolution/discrétisation variable**.

Remark : Si la discrétisation est coonvergente ce qui est appris a une ou plusieurs résolutions (si elles sont assez fine) sera valable pour une nouvelle résolution a erreur de discrétisation pret.

Apprentissage

- On se donne un modèle paramétrique $\mathcal{G}_{\boldsymbol{\theta}}$
- On se donne une distribution de probabilité $\mathrm{d}\mathbb{P}(\mathbf{a})$ sur un sous ensemble $\mathcal{A}$ de fonctions de $\mathcal{H}$ (par exemple $\mathrm{L}^2(\Omega)$). Notre objectif est de minimiser la perte

$$\mathcal{L}(\boldsymbol{\theta}) = \int_{\mathcal{A}} \int_{\Omega} \| \mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a}) - \mathcal{G}(\mathbf{a}) \|^2 \, \mathrm{d}\mathbf{x}\mathrm{d}\mathbb{P}(\mathbf{a})$$

- Evidemment en pratique on minimise une version discrétisée de cette perte. On se donne un ensemble d'entrainement $\mathcal{S} = \{(\mathbf{a}_k, \mathbf{u}_k) = (\mathbf{a}_k, \mathcal{G}_{\mathrm{ref}}(\mathbf{u}_k)), k = 1, ..., N\}$ avec $\mathbf{a}_k$ des fonctions et $\mathbf{u}_k$ leur image par l'opérateur qu'on cherche à approcher. On minimise la perte empirique

$$\mathcal{L}^h(\boldsymbol{\theta}) = \frac{1}{K} \sum_{k=1}^{K} \| \mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a}_k) - \mathbf{u}_k \|_{\mathcal{H}}^2 = \frac{1}{K} \sum_{k=1}^{K} \sum_{j=1}^{M} \| \mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a}_k)(\mathbf{x}_i) - \mathbf{u}_k(\mathbf{x}_i) \|^2$$

Remark: Puisqu'ici notre objectif est d'apprendre un opérateur qui en gros est l'opérateur inverse de notre EDP on sait que la sortie du modèle doit être solution de l'EDP. On peut donc ajouter des termes de régularisation qui seront les fonctions de coût associées aux résidus de l'EDP.

Apprentissage physiquement informés

Si on considère une EDP général de la forme:

$$\begin{cases} \mathcal{L}(\mathbf{u}, \mathbf{b}) = \mathbf{f}(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega \\ \mathcal{B}(\mathbf{u}) = \mathbf{g}(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\Omega \end{cases}$$

avec $\mathbf{b}$ les fonctions paramètres de l'EDP. On nomme $\mathbf{a}(\mathbf{x}) = (\mathbf{b}(\mathbf{x}), \mathbf{f}(\mathbf{x}), \mathbf{g}(\mathbf{x}))$ les entrées du problèmes. Alors on peut ajouter une régularisation qui en continun sera de la forme:

$$\mathrm{L}_{\mathrm{phy}}(\boldsymbol{\theta}) = \int_{\mathrm{A}} \int_{\Omega} \parallel \mathcal{L}(\mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a}), \mathbf{b}) - \mathbf{f} \parallel^2 \mathrm{d}\mathbf{x}\mathrm{d}\mathbb{P}(\mathbf{a}) + \int_{\mathrm{A}} \int_{\partial\Omega} \parallel \mathcal{B}(\mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a})) - \mathbf{g} \parallel^2 \mathrm{d}\mathbf{x}\mathrm{d}\mathbb{P}(\mathbf{a})$$

Une fois dicrétisée on minimisera

$$\mathrm{L}^{\mathrm{h}}_{\mathrm{phy}}(\boldsymbol{\theta}) = \frac{1}{\mathrm{K}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \sum_{\mathrm{j}=1}^{\mathrm{M}} \parallel \mathcal{L}(\mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a}_{\mathrm{k}}), \mathbf{b}_{\mathrm{k}})(\mathbf{x}_{\mathrm{j}}) - \mathbf{f}_{\mathrm{k}}(\mathbf{x}_{\mathrm{j}}) \parallel^2 + \frac{1}{\mathrm{K}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \sum_{\mathrm{j}=1}^{\mathrm{M}'} \parallel \mathcal{B}(\mathcal{G}_{\boldsymbol{\theta}}(\mathbf{a}_{\mathrm{k}}))(\mathbf{x}_{\mathrm{j}'}) - \mathbf{g}_{\mathrm{k}}(\mathbf{x}_{\mathrm{j}}) \parallel^2$$

Remark: Les apprentissage physique sans données sont très dur. En pratique on utilise les fonctions de coûts physique pour régulariser.

2. First architectures: CNN

Contexte

- l'architecture CNN vient de la vision par ordinateur et a été pensé pour les problèmes d'imagerie.
- Elle rentre dans le cadre précédent.

Objectives:
- Voir comment les spécificités des problèmes de vision permettent de justifier des choix de noyaux et des architectures
- Etudiez les discrétisations utilisées

- Une image en noir et blanc = discrétisation sur grille regulière d'une fonction $v : \mathbb{R}^2 \to \mathbb{R}$.
- Une image en couleur = discétisation sur grille regulière d'une fonction de $\mathbf{v} : \mathbb{R}^2 \to \mathbb{R}^3$.
- On appelle la discrétisation $v_h \in \mathbb{R}^{n,n,q}$ avec q le nombre de canneaux (1 pour noir et blanc, 3 pour couleur, plus si plusieurs images).

Remark: Puisqu'il s'agit de prendre en entrée des fonctions on peut utiliser les architectures génériques précédente.

Contexte suite

- la classification ou l'on veut construire

$$\Phi(\mathrm{v}) \to \mathrm{y} \in \{1, ..., \mathrm{C}\}$$

- Ici y represente le numéro de la classe qui est un entier.
- la segmentation ou l'on veut construire

$$\Phi(\mathrm{v})(\mathbf{x}) \to \mathrm{w}(\mathbf{x}) \in [0, 1]^{\mathrm{n},\mathrm{n},\mathrm{q}}$$

avec des pixel qui vaille zero ou un selon si on est sur le contour de l'objet ou pas.

Difficulties : Comment choisir la paramétrisation de nos noyaux, nos couches et la discrétisation ?

Idea (A priori) : Faire les choix d'architectures en fonction d' apriori qu'on a sur l'opérateur qu'on souhaite approcher.

Symétrie

Definition (Groupe): Un groupe est un ensemble G muni d'une loi de composition interne $\circ : G \times G \to G$ qui satisfait les propriétés suivantes:

- (associativité) $\forall g_1, g_2, g_3 \in G, (g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$
- (élément neutre) $\exists e \in G, \forall g \in G, e \circ g = g \circ e = g$
- (élément inverse) $\forall g \in G, \exists g^{-1} \in G, g \circ g^{-1} = g^{-1} \circ g = e$

Definition (Action de Groupe): Soit G un groupe et X un ensemble. On appelle action de groupe une application $\cdot : G \times X \to X$ qui satisfait les propriétés suivantes:

- (élément neutre) $\forall x \in X, e \cdot x = x$
- (compatibilité) $\forall g_1, g_2 \in G, \forall x \in X, \underbrace{(g_1 \circ g_2)}_{\in \mathcal{G}} \cdot x = g_1 \cdot \underbrace{(g_2 \cdot x)}_{\in X}$

Remark (actions de groupe): elles formalise l'analyse d'un ensemble de transformations qui vont agir sur un ensemble X

Symétrie 2

- Exemple 1:
  - $X = \mathbb{R}^2$ et $G = \left\{A \in \mathcal{M}_{2,2} \mid \| \mathbf{x} \|_2 = \| A\mathbf{x} \|, \forall \mathbf{x} \in X\right\}$ muni de la **composition**
  - En pratique G matrice de rotation de déterminant positif et négatif.
  - **Action**: $g \cdot \mathbf{x} = A\mathbf{x}$
  - Appliquer une première rotation puis une autre est équivalent à appliquer la composition de ses rotations. On a donc une **action de groupe**
- Exemple 2:
  - $X = \mathbb{R}$, $G = \mathbb{Z}$ muni de l'addition
  - **action**: $n \cdot x = x + n$
  - additionner n à x puis m est équivalent a additionner $n + m$

Proposition: Soit f une application $f : X \to X$ et G un groupe qui agit sur X. L'application suivante

$$(g \cdot f)(x) = f\left(g^{-1} \cdot x\right)$$

est une action de groupe sur l'ensemble des endormorphismes de X dans $\mathbb{X}$.

- permet d'étudier l'effet d'un ensemble de transformation agissant sur les entrées d'une fonction modifie la fonction.

Invariance/equivariance

Definition (Invariance de groupe) : Soit G un groupe qui agit sur Ω. On dit qu'une application $f : X \to X$ est invariante par l'action de G si

$$\forall g \in G, \forall f \in L(X, X), f(g \cdot x) = f(x)$$

Definition (Equivariance de groupe) : Soit G un groupe qui agit sur X. On dit qu'une application $f : X \to X$ est équivariante par l'action de G si

$$\forall g \in G, \forall f \in L(X, X), f(g \cdot x) = g \cdot f(x)$$

- L'invariance de groupe revient à dire que l'application f ne dépend pas de la transformation de l'entrée par le groupe.
- L'équivarance revient à dire que la transformation de l'entrée par le groupe se répercute sur la sortie par la même transformation.

Image et invariance

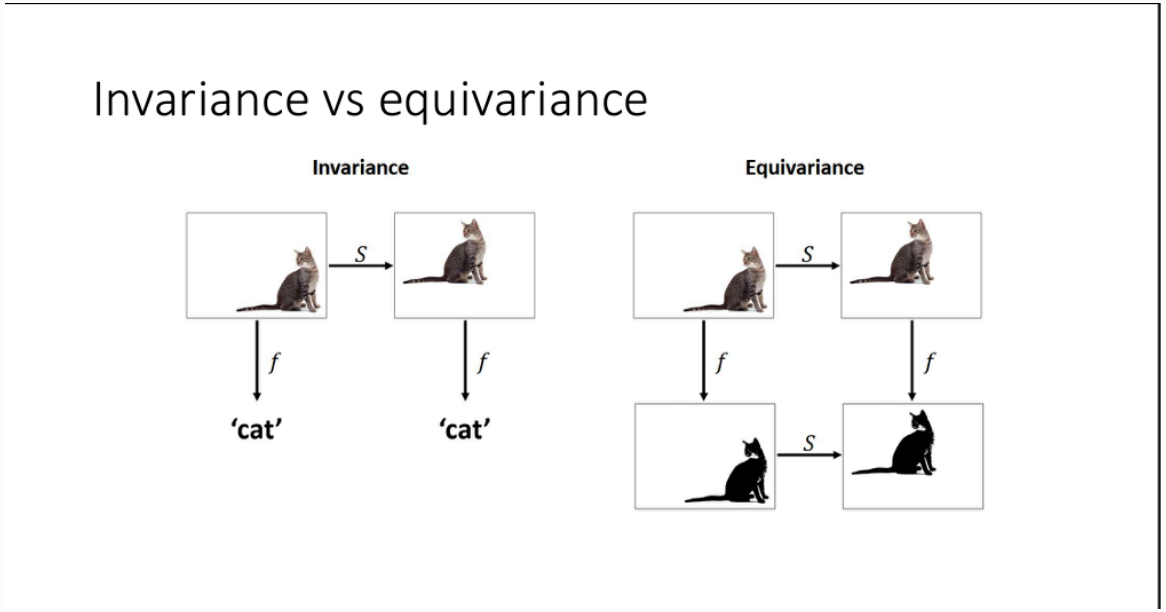


Figure 1: A curious figure.

- La classfication d'image est une opération **invariante par translation**
- la segmentation d'image est une opération **équivariente par translation**.

Objectives : Introduire cette structure dans les réseaux agissant sur des images.

Convolution et invariance

Lemma (Caractérisation de operateurs a noyau invariant par translation): Soit $\mathcal{B} : H \to H$ un opérateur à noyau linéaire et continu de la forme

$$\mathcal{B}(f)(\mathbf{x}) = \int_{\mathbb{R}^d} k(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) d\mathbf{y}$$

On suppose que $\mathcal{B}$ est équivariante par le groupe des translations. Alors il existe une fonction $k_c : \mathbb{R}^d \to \mathbb{R}$ telle que

$$k(\mathbf{x}, \mathbf{y}) = k_c(\mathbf{x} - \mathbf{y})$$

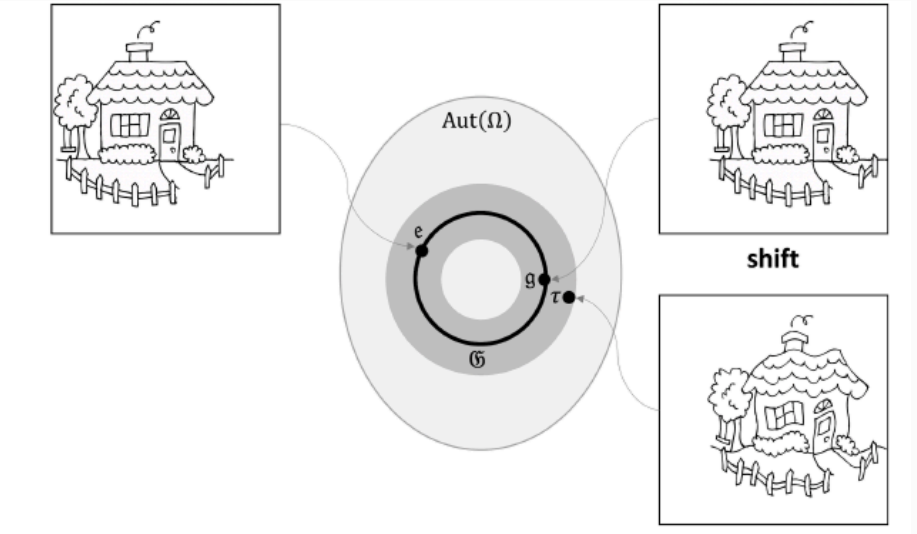
pour tout $f \in H$.

- Preuve en cours

Remark: Par conséquent pour des problèmes de vision on prend des **noyaux de type convolution**

Stabilité aux petites deformations

- Si on applique une petite déformation au signal d'entrée on s'attend à ce que la sortie soit peu ou pas modifiée. Si on reprend l'exemple de l'image du chat, une légère dilatation implique un léger changement sur le résultat d'une segmentation par exemple.



Definition (Stabilité aux petites déformations): Soit un signal $\Phi(f) \in L(H, H)$ une transformation sur les signaux f. Cette transformation est dite stable aux déformations si

$$\| \Phi(\tau \cdot f) - \Phi(f) \| \leq Cc(\tau) \| f \|$$

On parle de stabilité aux petites déformations si $c(\tau) = O(\varepsilon)$ avec C une constante indépendante de ε.

Stabilité aux petites deformations

Remark : Des analyses de Fourier nous montre qu'il faut que les convolutions soit localisées en espace.

- Forme de notre couche non-locale:

Definition (Couche de convolution spatiale continue): Soit $\mathbf{v}_l : \mathbb{R}^d \to \mathbb{R}^\in$ et $\mathbf{v}_{l+1} : \mathbb{R}^d \to \mathbb{R}^\in$. On appelle une couche de convolution spatiale l'opérateur

$$\mathbf{v}_{l+1}(\mathbf{x}) = \mathcal{B}_{\boldsymbol{\theta}}(\mathbf{v})(\mathbf{x}) = \int_{B(\mathbf{x},r)} k_\theta(\mathbf{x}-\mathbf{y})\mathbf{v}_l(\mathbf{y})d\mathbf{y}$$

ou $k_\theta : \mathbb{R}^d \to \mathbb{R}^{out\times in}$ est la partie apprenable.

Couche de convolutions discretes

Definition (Couche de convolution 1D spatiale discrète): Soit $\mathbf{v}_l \in \mathbb{R}^{n,in}$ et $\mathbf{v}_{l+1} \in \mathbb{R}^{n,out}$. On appelle une couche de convolution spatiale discrète l'opérateur

$$\mathbf{v}_{l+1}(i) = \frac{1}{(2s+1)} \sum_{j=-s}^{s} k_j \mathbf{v}_l(j)$$

ou $k \in \mathbb{R}^{2s+1}$ est la partie apprenable.

Definition (Couche de convolution 2D spatiale discrète): Soit $\mathbf{v}_l \in \mathbb{R}^{n,n,\in}$ et $\mathbf{v}_{l+1} \in \mathbb{R}^{n,n,\in}$. On appelle une couche de convolution spatiale discrète l'opérateur

$$\mathbf{v}_{l+1}(i,j) = \frac{1}{(2s+1)^2} \sum_{i',j'=-s}^{s} k_{i',j'} \mathbf{v}_l(i',j')$$

ou $k_{i',j'} \in \mathbb{R}^{(2s+1)\times(2s+1)}$ est la partie apprenable.

Aspects multi-échelles

- Les réseaux convolutifs utilise des aspects multi-échelles afin de pouvoir capter cas ou les échelles grossière ou locales domine

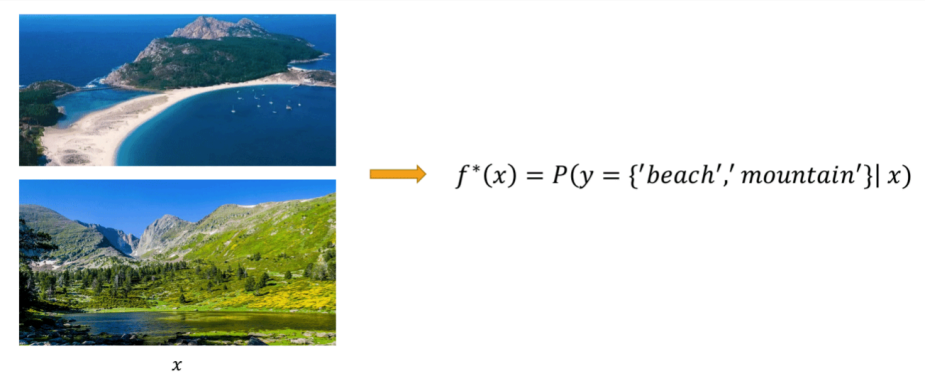


Figure 3: Classification on fine pictures

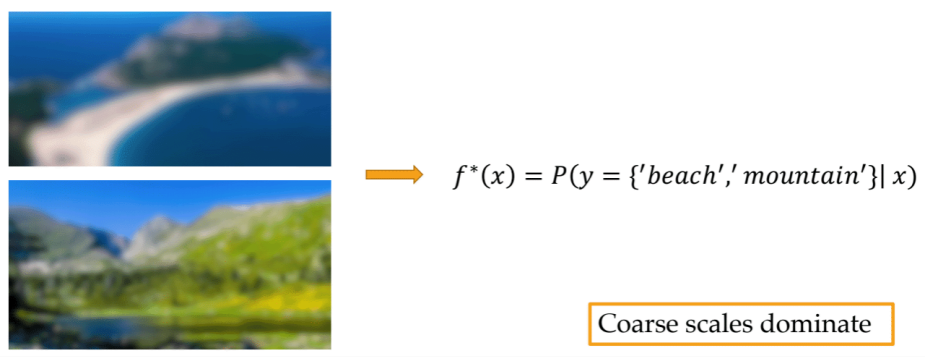


Figure 4: Classification on coarse pictures.

Aspects multi-échelles II

- Une seconde possbilité correspond à l'image Figure 5.

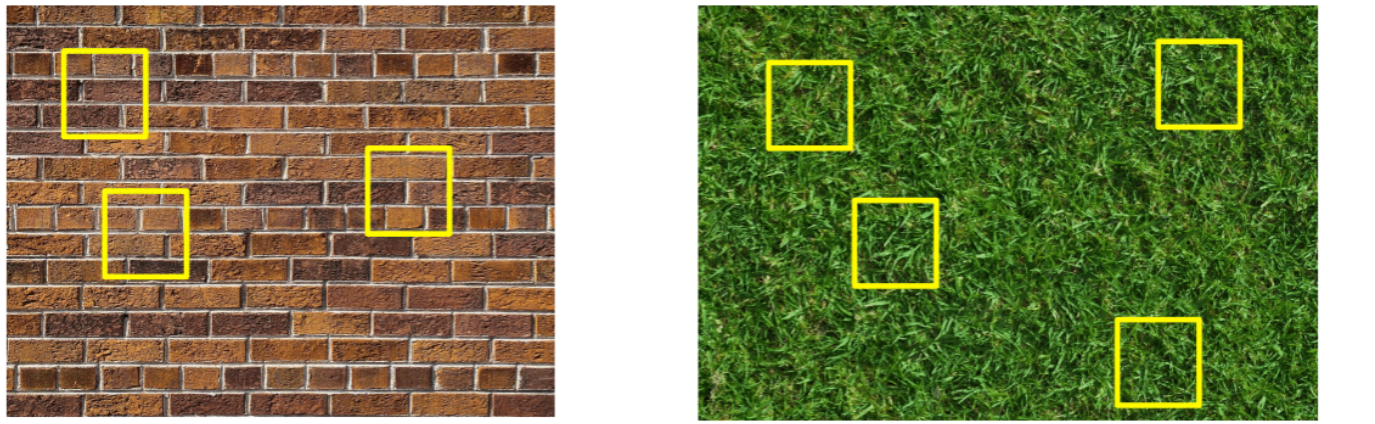


Figure 5: A curious figure.

- Ici on voit que la fonction qu'on cherche a construire $f(x) = \sum_{i=1} g(x_i)$ ou x_i est un petit patche de l'image d'origine.
- L'information d'un patch local est suffisante pour déterminer la classe de l'image.

Remark: On veut donc des architectures qui captent l'information locale et globale séparement et de les agréger.

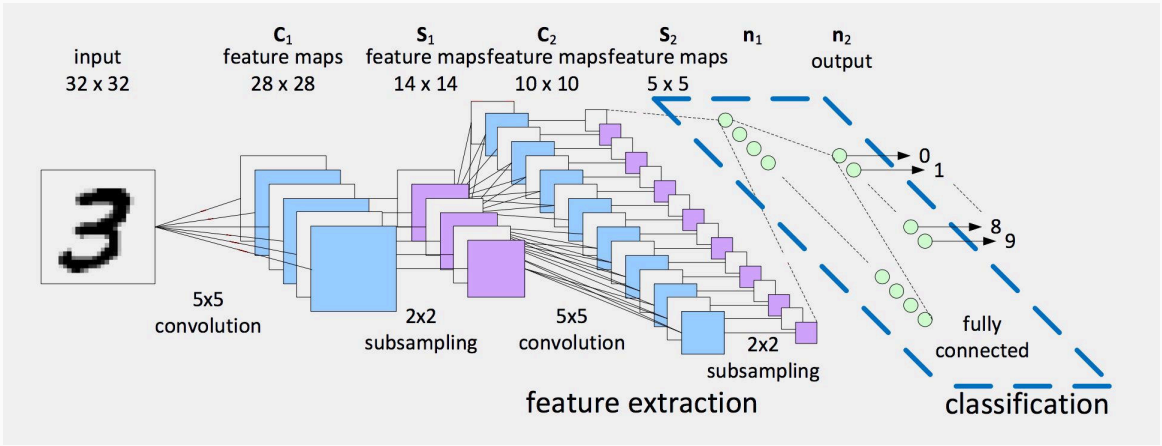
Pooling

- On veut traiter d'abord les interactions locales puis d'aller vers des interactions de plus en plus globales.
- Réduction de la dimension des images avec un opérateur de "pooling" qui dé-raffine l'image ou le signal.

Definition (Couche de mean-pooling en 2D): Soit $\mathbf{v}_l \in \mathbb{R}^{n,n,\in}$ et $\mathbf{v}_{l+1} \in \mathbb{R}^{\frac{n}{2},\frac{n}{2},in}$. On appelle une couche de mean-pooling l'opérateur

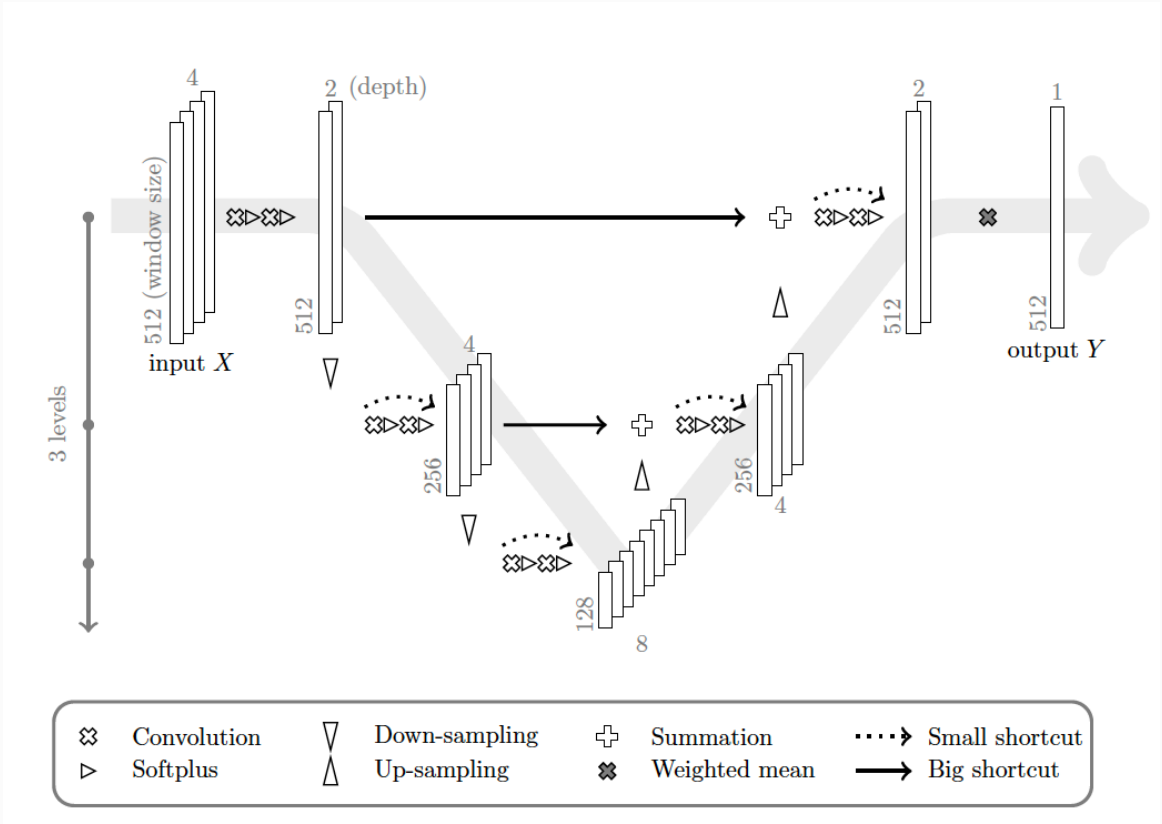
$$\mathbf{v}_{l+1}(i,j) = \frac{1}{4}(\mathbf{v}_l(2i,2j) + \mathbf{v}_l(2i+1,2j) + \mathbf{v}_l(2i,2j+1) + \mathbf{v}_l(2i+1,2j+1))$$

- Architecture pour la classification:



Architecture Unet

- Pour la segmentation on va alterner des couches de convolution et de pooling pour réduire la dimension puis des couches de up-pooling et de convolution pour remonter en dimension



- Pour les EDP il est assez naturel d'utiliser cette architecture.

CNN et NO

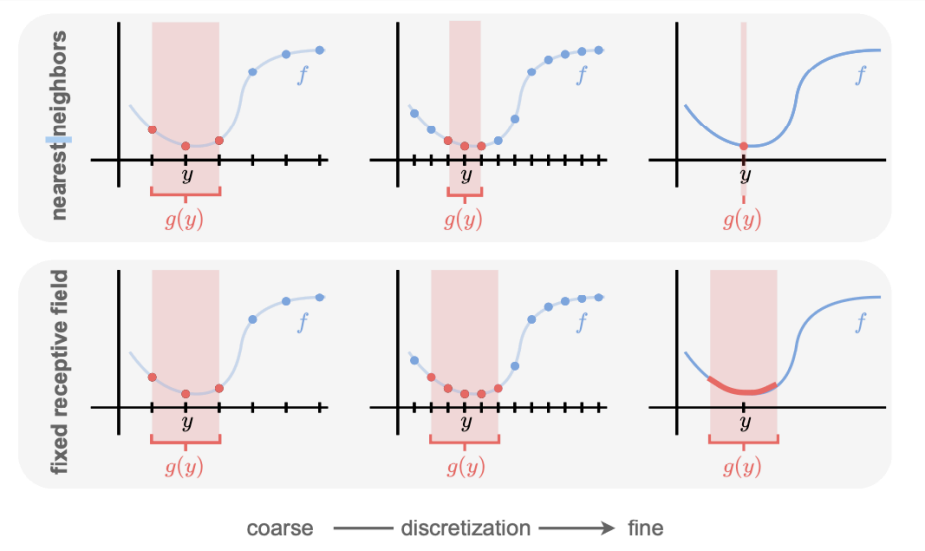
- **Est ce que les CNN sont des opérateurs neuronaux ?**: NON
- une couche de convolution est discrétisation sur grille uniforme de l'opérateur:

$$\int_{B(x,2sh)} k_\theta(x-y)v_l(y)dy$$

avec s la taille du noyau en nombre de pixel et h le pas de la grille. On a

$$\lim_{h\to\infty} \int_{B(x,2sh)} k_\theta(x-y)v_l(y)dy = \int_{\mathbb{R}^d} \delta(x-y)v_l(y)dy = v_l(x)$$

On voit qu'on ne converge pas du tout vers l'opérateur cible qui était $\int_{B(x,r)} k_\theta(x-y)v_l(y)dy$ avec r fixé.



3. Kernel based architectures

GreenNet

- Etant donné que la théorie de Green nous dit que pour des EDP linéaires l'opérateur inverse est un opérateur à noyau. Il est donc naturel de chercher a construire des architectures qui vont mimer cela.

Definition (Réseau de neurones GreenNet continue): Soit $\mathbf{f} : \mathbb{R}^{d} \to \mathbb{R}^{in}$ et $\mathbf{u} : \mathbb{R}^{d} \to \mathbb{R}^{out}$. L'opérateur GreenNet est défini par

$$\mathbf{u}(\mathbf{x}) = \int_{\Omega} k_{\theta}(\mathbf{x}, \mathbf{y})\mathbf{f}(\mathbf{y})d\mathbf{y} + \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x})$$

ou $k_{\theta} : \Omega \times \Omega \to \mathbb{R}^{out \times in}$ et $\mathbf{h}_{bold} : \Omega \to \mathbb{R}^{\in}$ sont des réseaux de type MLP.

- Le second réseau à pour but d'apprendre partie homogène liée aux domaine et aux conditions aux limites et au domaine.

Definition (Réseau de neurone GreenNet): Soit $\mathbf{f} \in \mathbb{R}^{n,in}$ et $\mathbf{u} \in \mathbb{R}^{n,out}$. L'opérateur GreenNet discrétisé est défini par

$$\mathbf{u}(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} k_{\theta}(\mathbf{x}_i, \mathbf{x}_j)\mathbf{f}(\mathbf{x}_j) + \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

ou $k_{\theta} : \Omega \times \Omega \to \mathbb{R}^{out \times in}$ et $\mathbf{h}_{bold} : \Omega \to \mathbb{R}^{out}$ sont des réseaux de type MLP.

GreenNet et NO

- NO ou non NO ?
- Si on considère que les points $\mathbf{x}_i$ sont distribués selon une loi de densité $p(\mathbf{x})$ la méthode de Monte Carlo nous dit que:

$$\frac{1}{n}\sum_{j=1}^{n} k_\theta(\mathbf{x}_i, \mathbf{x}_j)\mathbf{f}(\mathbf{x}_j) \approx \mathbb{E}_{\mathbf{x}\sim p}[k(\mathbf{x}, \mathbf{y})f(\mathbf{y})] = \int_\Omega k_\theta(\mathbf{x}_i, \mathbf{y})\mathbf{f}(\mathbf{y})p(\mathbf{y})d\mathbf{y}$$

- Donc notre couche discrétisée converge vers le bonne opérateur que si la densité $p(\mathbf{y})$ est uniforme.
- On en déduit que cette architecture n'est pas un opérateur neuronal tel quel.

Definition (Opérateur neuronal GreenNet): Soit $\mathbf{f} \in \mathbb{R}^{n,\in}$ et $\mathbf{u} \in \mathbb{R}^{n,\in}$. L'opérateur GreenNet discrétisé avec pondération est défini par

$$\mathbf{u}(\mathbf{x}_i) = \frac{1}{n}\sum_{j=1}^{n} \frac{k_\theta(\mathbf{x}_i, \mathbf{x}_j)}{p(\mathbf{x}_j)}\mathbf{f}(\mathbf{x}_j) + \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

ou $k_\theta : \Omega \times \Omega \to \mathbb{R}^{\in\times\in}$ et $\mathbf{h}_{\texttt{bold}} : \Omega \to \mathbb{R}^{\in}$ sont des réseaux de type MLP et $p(\mathbf{x})$ est la densité des points $\mathbf{x}_i$.

GreenNet et NO

- En effet, par les estimation de Monte Carlo on a

$$\frac{1}{n}\sum_{j=1}^{n}\frac{k_\theta(x_i,x_j)}{p(x_j)}f(x_j)\approx\mathbb{E}_{x\sim p}\left[\frac{k(x,y)}{p(y)}f(y)\right]=\int_\Omega k_\theta(x_i,y)f(y)dy$$

ce qui est bien l'opérateur qu'on cherche a approcher.

Remark: Si on se réfere a des exemples comme le Laplacien sur un domaine simple comme le carré on sait que le noyau de Green est singulier en $x=y$.

- On peut se demander si le réseau de neurones est capable d'apprendre cette singularité. Une des méthodes proposés dans la littératues est l'utilisation de fonctions d'activation rationnelle.

Definition (Fonction d'activation rationnelle): On appelle fonction d'activation rationnelle de Padé:

$$\sigma(x)=\frac{\sum_{i=0}^{p}a_i x^i}{1+|\sum_{j=0}^{q}b_j x^j|}$$

avec a_i, b_j des paramètres apprenables. En général on prend $p=3$ et $q=2$.

Approximation de bas rang I

- Problème de complexité et de mémoire. L'évaluation d'une couche de GreenNet est en $O(n^2)$.

Definition (Rang d'une fonction bivarié): Soit $k : \Omega \times \Omega \to \mathbb{R}$ une fonction bivariée. On dit que k est de rang r si elle peut s'écrire

$$k(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{r} \varphi_j(\mathbf{x})\psi_j(\mathbf{y})$$

avec $\varphi_j, \psi_j : \Omega \to \mathbb{R}$.

- On voit bien que si on a une telle décomposition existe on peut evaluer de tel fonction avec un cout $O(nr)$ au lieu de $O(n^2)$.
- Est ce qu'on peut faire ce genre de décomposition ?

Theorem (Théorème de Schmidt): Soit $k \in L^2(\Omega \times \Omega)$ une fonction bivariée. Il existe des suites orthonormées $(\varphi_j)_{jg.=1}$ et $(\psi_j)_{j\geq 1}$ dans $L^2(\Omega)$ et une suite décroissante de réels positifs $(\sigma_j)_{j\geq 1}$ telle que

$$k(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{\infty} \sigma_j \varphi_j(\mathbf{x})\psi_j(\mathbf{y})$$

avec la convergence dans $L^2(\Omega \times \Omega)$.

Approximation de bas rang II

Proposition : Soit $k \in L^2(\Omega \times \Omega)$ une fonction bivariée. On note

$$k_r(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{r} \sigma_j \varphi_j(\mathbf{x}) \psi_j(\mathbf{y})$$

la troncature de rang r de k. On a

$$\| k - k_r \|_{L^2(\Omega \times \Omega)} = \sqrt{\sum_{j=r+1}^{\infty} \sigma_j^2}$$

- On ne detaillera pas les arguments mais si vous dicrétisez votre integrale avec une méthode de quadrature convergente vous avez

$$\int_{\Omega} k(\mathbf{x}, \mathbf{y}) \mathbf{f}(\mathbf{y}) d\mathbf{y} \approx \frac{1}{n} \sum_{j=1}^{n} \frac{k(\mathbf{x}, \mathbf{x}_j)}{p(\mathbf{x}_i)} \mathbf{f}(\mathbf{x}_j) = K_n \mathbf{f}$$

avec $K_{n,i,j} = \frac{k(\mathbf{x}, \mathbf{x}_j)}{p(\mathbf{x}_i)}$ vous aller avoir un résultat du type

$$\sigma, \varphi, \psi = \lim_{n \to \infty} U_n, V_n, \Sigma_n$$

avec $U_n, V_n, \Sigma_n = \mathrm{svd}(K_n)$

Idea : Faire une approximation de bas rang de la matrice K_n pour approcher l'opérateur inverse.

Approximation de bas rang III

Definition (Opérateur neuronale GreenNet bas rang): Soit $\mathbf{f} \in \mathbb{R}^{n,\in}$ et $\mathbf{u} \in \mathbb{R}^{n,\in}$. L'opérateur GreenNet bas rang est défini par

$$\mathbf{u}(\mathbf{x}_i) = \sum_{j=1}^{r} \langle \mathbf{f}, \psi_{\boldsymbol{\theta},j} \rangle \varphi_{\boldsymbol{\theta},j}(\mathbf{x}_i) + \mathbf{h}_{\boldsymbol{\theta}}(\mathbf{x}_i)$$

ou $\varphi_{\boldsymbol{\theta}}, \psi_{\boldsymbol{\theta}} : \Omega \to \mathbb{R}$ et $\mathbf{h}_{\mathtt{bold}} : \Omega \to \mathbb{R}^{\in}$ sont des réseaux de type MLP et

$$\langle \mathbf{f}, \psi_{\boldsymbol{\theta}_j} \rangle = \frac{1}{n} \sum_{j=1}^{n} \frac{1}{p(\mathbf{x}_j)} \mathbf{f}(\mathbf{y}_j) \psi_{\boldsymbol{\theta},j}(\mathbf{y}_j)$$

- On voit que contrainrement au GreenNet classique le coût de cette couche est $O(nr)$ avec r le rang de l'approximation. Autre approximation appelée **DeepOnet**.
- On repart de la couche continue GreenNet bas rang

$$\mathbf{u}(\mathbf{x}) = \sum_{j=1}^{r} \langle \mathbf{f}, \psi_{\boldsymbol{\theta},j} \rangle \varphi_{\boldsymbol{\theta},j}(\mathbf{x})$$

- La partie la plus couteuse est le calcul des r intégrales. Vu que les fonctions $\psi_{\boldsymbol{\theta},j}$ sont apprises on ne le connait pas a l'avance.

DeepONet

Idea: On a juste besoin des modes $\langle \mathbf{f}, \psi_{\boldsymbol{\theta},j} \rangle$ et pas la fonction entière une naturel est essayer de directement apprendre la valeur de ses modes.

- Pour cela on se donne des points $\mathbf{y}_1, ..., \mathbf{y}_m$ dans Ω qu'on appelera des senseurs. ON suppose qu'on peut connaitre les fonctions $\mathbf{f}$ a ses points. La méthode deepOnet va proposé un réseau appeler BranchNet:

$$B_{\boldsymbol{\theta}}(\mathbf{y}_1, ..., \mathbf{y}_m) = (c_1, ...c_r)$$

Definition (Opérateur neuronal DeepOnet): Soit $\mathbf{f} \in \mathbb{R}^{m,\in}$ et $\mathbf{u} \in \mathbb{R}^{n,\in}$. L'opérateur DeepOnet est défini par

$$\mathbf{u}(\mathbf{x}_i) = \sum_{j=1}^{r} c_j \varphi_{\boldsymbol{\theta},j}(\mathbf{x}_i)$$

ou $\varphi_{\boldsymbol{\theta}} : \Omega \to \mathbb{R}$ est un réseau de type MLP appelé TrunkNet et

$$(c_1, ..., c_r) = B_{\boldsymbol{\theta}}(\mathbf{f}(\mathbf{y}_1), ..., \mathbf{f}(\mathbf{y}_m))$$

avec $\mathbf{y}_1, ..., \mathbf{y}_m$ des points dans Ω. $B_{\boldsymbol{\theta}}$ le branchNet est aussi un MLP.

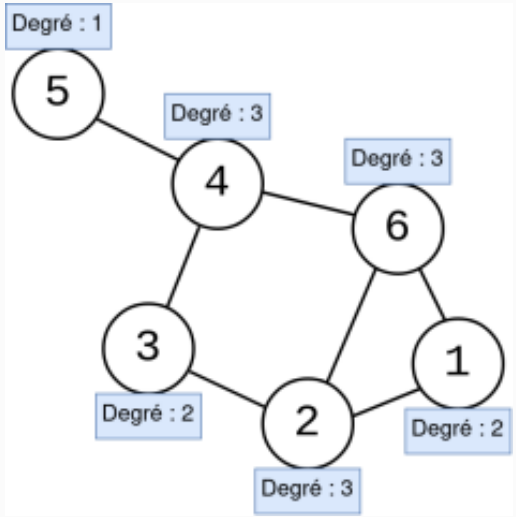
4. Spatial GNN

Graphes

- Les architectures CNN supposent que les données sont sur une grille cartésienne.
- Les architectures basées sur les noyaux supposent que les données sont sur des points distribués dans le domaine.
- **Next**: Graphes et maillages.

Definition (Graphe): Soit V un nombre fini de sommets. Un **graphe** est définie par une paire $G = [V, E]$ avec V l'ensembles des noeuds et E l'ensemble des arêtes ou un élément de E est une paire d'élément de V noté (x, y) avec $x, y \in V$.

Definition (Degré d'un sommet): Soit $x \in V$ un sommet on nomme d_x le degré de x le nombre d'arêtes qui connecte x à d'autre noeud. En général on note $\Delta(G)$ et $\delta(G)$ les degrés maximal et minimal. On parle d'un graphe régulier si $\Delta(G) = \delta(G)$.



Graphes 2

Definition (Voisinage d'un sommet): Soit $x \in V$ un sommet on nomme le voisinage $N(x)$ l'ensemble de sommets connectés à x par une arête.

Definition (Matrice d'adjacence): Soit $G = [V, E]$ un graphe. La matrice d'adjacence est une matrice $A \in \mathcal{M}_{n,n}(\mathbb{R})$ tel que

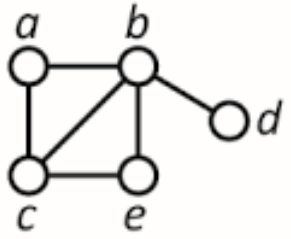
$$A_{ij} = \begin{cases} 1 & \text{si} \quad (x_i, x_j) \in E \\ \\ 0 & \text{else} \end{cases}$$

On peut aussi nommé le graphe $G = [V, A]$.

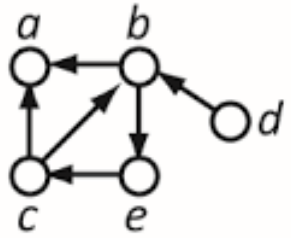
Definition (Orientation de graphe):

- Graphe **non orienté/non orienté** et donc

$$A_{ij} = A_{ji} = 1; \quad A_{ij} = 1, A_{ji} = 0$$



un graphe non orienté



un graphe orienté

Graphe 3

Definition (Graphe pondéré): Soit $G = [V, A]$ un graphe. Le graphe est pondéré si il est munit d'une matrice W tel que

$$W_{ij} = \begin{cases} w_{ij} & \text{si} \quad (x_i, x_j) \in E \\ 0 & \text{else} \end{cases}$$

On peut le noter $G = [V, W]$.

Definition (Volume d'un noeud): Soit $G = [V, W]$ un graphe pondéré. On définit le volume (notion qui remplace le degré) d'un noeud comme

$$d_i = \sum_{j \in N(i)} w_{ij}$$

Graphe 4

Definition (Distance entre 2 points): Soit $G = [V, A]$ un graphe. On définit un chemin entre x et y par:

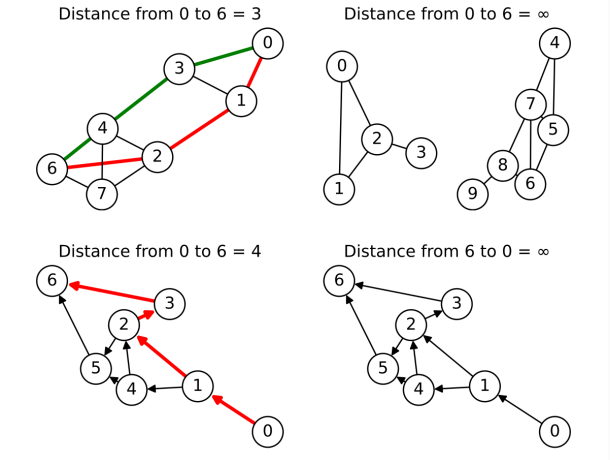
$$\gamma(x, y) = ((x, z_1), (z_1, z_2), ..., (z_n, y))$$

une succéssion d'arête qui relie les deux point. On définit la distance par

$$dis(x, y) = \min_{\gamma} | \gamma |$$

avec $| \gamma | = \sum_{i \in \gamma} w_{ij}$

Definition (Graphe acyclique): On parle de graphe acyclique si pour tous les noeuds du graphe il n'existe aucun chemin entre le noeud et lui même.



Fonctions sur des graphes

Definition (Espace de Hilbert des sommets): On note $\mathcal{H}(V, \chi)$ l'espace des fonctions définit sur les sommets. Il s'agit d'un espace de Hilbert $(\mathbb{R}^n, \langle ., . \rangle_v)$,avec $n = \mid V \mid$, muni du produit scalaire

$$\langle f, g \rangle_V = \frac{1}{n} \sum_{i=1}^{n} f_i g_i \chi_i$$

avec $\chi_i = \chi_{in}\left(\frac{1}{n}\sum_{j=1}^{n} w_{ji}\right) + \chi_{out}\left(\frac{1}{n}\sum_{j=1}^{n} w_{ij}\right)$ (la somme des poids des arrêtes entrantes et sortantes) et avec $\chi_{in}(x) > 0, \chi_{out}(x) > 0$ sur $\mathbb{R}_+^*$ et $\chi_{in}(0) = 0, \chi_{out}(0) = 0$.

Definition (Espace de Hilbert des arêtes): On note $\mathcal{H}(E, \psi)$ l'espace des fonctions définit les arêtes. Il s'agit d'un espace de Hilbert $\left(\mathbb{R}^{2n^2}, \langle ., . \rangle_E\right)$ muni du produit scalaire

$$\langle f, g \rangle_E = \frac{1}{2n^2} \sum_{i,j=1}^{n} f_{ij} g_{ij} \psi\left(w_{ij}\right)$$

avec $\psi(x) > 0$ sur $\mathbb{R}_+^*$ et $\psi(0) = 0$.

GNN

Definition (GNN): Soit $\mathrm{G} = [\mathrm{V}, \mathrm{W}]$ un graphe pondéré. Un GNN est une application paramétrique

$$\Phi_{\boldsymbol{\theta}} : \mathcal{M}_{\mathrm{n,n}} \times \mathcal{H}(\mathrm{V}, \chi) \times \mathcal{H}(\mathrm{E}, \psi) \to \mathcal{M}_{\mathrm{n,n}} \times \mathcal{H}(\mathrm{V}, \chi) \times \mathcal{H}(\mathrm{E}, \psi)$$

qui prend donc en entrée la matrice d'adjacence, une fonction sur les noeuds et une fonction sur les arêtes et qui renvoit la même chose.

Definition (Message and aggregation GNN sublayer): Soit $\mathrm{G} = [\mathrm{V}, \mathrm{W}]$ un graphe pondéré. Soit $\mathbf{f} \in \mathrm{H}(\mathrm{V}, \chi)$ et $\mathbf{e} \in \mathrm{H}(\mathrm{E}, \psi)$. On appelle étape de message et d'agrégation l'opération qui pour chaque noeud $\mathrm{v_i}$ calcule

$$\mathbf{m}_{\mathrm{i}} = \sum_{\mathrm{j} \in \mathrm{N(i)}} \frac{\mathrm{w_{ij}}}{\mathrm{d_i}} \varphi_{\boldsymbol{\theta}}\left(\mathbf{h}_{\mathrm{i}}, \mathbf{h}_{\mathrm{j}}, \mathbf{e}_{\mathrm{ij}}\right)$$

avec $\mathbf{h}_{\mathrm{i}}$ la valeur des fonctions $\mathbf{h}$ au noeud i, $\mathbf{e}_{\mathrm{ij}}$ la valeur de la fonction sur l'arête (i, j) et $\varphi_{\boldsymbol{\theta}} : \mathbb{R}^{2\mathrm{in}+\mathrm{e}} \to \mathbb{R}^{\mathrm{m}}$ (MLP).

GNN II

Definition (Update step GNN sublayer): Soit $G = [V, W]$ un graphe pondéré. Soit $\mathbf{f} \in H(V, \chi)$ et $\mathbf{e} \in H(E, \psi)$. On appelle étape de mise à jour l'opération qui pour chaque noeud v_i calcule

$$\mathbf{h}'_i = \gamma_{\boldsymbol{\theta}}(\mathbf{h}_i, \mathbf{m}_i)$$

avec $\mathbf{h}_i$ la valeur des fonctions $\mathbf{h}$ au noeud i, $\mathbf{m}_i$ le message agrégé et $\gamma_{\boldsymbol{\theta}} : \mathbb{R}^{in+m} \to \mathbb{R}^{in}$ une fonction paramétrée par un réseau de neurones.

Definition (Message Passing (MS)): Soit $G = [V, W]$ un graphe pondéré. Soit $\mathbf{f} \in H(V, \chi)$ et $\mathbf{e} \in H(E, \psi)$. Une couche de **Message Passing** est définie par l'enchainement d'une étape de message et d'agrégation suivie d'une étape de mise à jour.

Remark: Pour faire le lien avec des opérateurs neuronaux il faut pouvoir passer à la limite dans les graphes et fonctions sur graphes.

GNN

Definition (Graphe discrétisant une variété): Soit $\mathcal{M} \subset \mathbb{R}^d$ une variété compacte. Soit $p(\mathbf{x})$ une densité de probabilité sur $\mathcal{M}$. On dit qu'une suite de graphe $(G_n)_{n\geq 1}$ discrétise $\mathcal{M}$ si pour tout n le graphe $G_n = [V_n, W_n]$ est tel que

- $V_n = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ avec $\mathbf{x}_i$ des échantillons indépendants de la loi de densité $p(\mathbf{x})$.
- $$\frac{w_{n,ij}}{d_i} = k(\mathbf{x}_i, \mathbf{x}_j) = \frac{W(\mathbf{x}_i, \mathbf{x}_j)}{\int_{\mathcal{M}} W(\mathbf{x}_i, \mathbf{y})d\mathbf{y}}$$

  avec $W : \mathcal{M} \times \mathcal{M} \to \mathbb{R}_+$ un noyau de pondération et d le degré du noeud i.

Definition (Continue version of the update MS step): Soit $\mathbf{f} : \mathcal{M} \to \mathbb{R}^{in}$ une fonction. On appelle couche locale l'opération

$$\mathbf{f}'(\mathbf{x}) = \gamma_{\boldsymbol{\theta}}(\mathbf{f}(\mathbf{x}), \mathbf{m}(\mathbf{x}))$$

avec $\gamma_{\boldsymbol{\theta}} : \mathbb{R}^{in} \to \mathbb{R}^{in}$ une fonction paramétrée par un réseau de neurones.

GNN

Definition (Continue version of the message and aggregation MS step): Soit $\mathbf{f} : \Omega \to \mathbb{R}^{\mathrm{in}}$ une fonction. On appelle couche non locale l'opération

$$\mathbf{m}(\mathbf{x}) = \int_{\mathrm{B}(0,\mathrm{r})\cap\mathcal{M}} \mathrm{k}(\mathbf{x},\mathbf{y})\varphi_{\boldsymbol{\theta}}(\mathbf{f}(\mathbf{x}),\mathbf{f}(\mathbf{y}),\mathbf{e}(\mathbf{x},\mathbf{y}))\mathrm{d}\mathbf{y}$$

avec $\mathrm{k} : \mathcal{M} \times \Omega \to \mathbb{R}$, un noyau de pondération et $\varphi_{\boldsymbol{\theta}} : \mathbb{R}^{2\mathrm{in}+\mathrm{e}} \to \mathbb{R}^{\mathrm{m}}$ une fonction paramétrée par un réseau de neurones.

Remark: La couche non locale tel qu'elle est définie ne définie par une couche d'opérateur neural pour deux raisons

- La couche qu'elle est définie ne tient pas compte de la densité de probabilité des points.
- La domaine d'intégration dépend de la résolution du graphe. En effet le voisinage est défini par les connections du graphe et non par en terme de distance dans l'espace physique comme les CNN.

GNN

Definition (Message and aggregation GNO sublayer): Soit $G = [V, W]$ un graphe pondéré. Soit $\mathbf{f} \in H(V, \chi)$ et $\mathbf{e} \in H(E, \psi)$. On appelle étape de message et d'agrégation l'opération qui pour chaque noeud v_i calcule

$$\mathbf{m}_i = \sum_{\mathbf{v}_j \in B(\mathbf{v}_i, r)} \frac{w_{ij}}{d(\mathbf{v}_i)p(\mathbf{v}_j)} \varphi_{\boldsymbol{\theta}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij})$$

avec $\mathbf{v}_i$ les sommets du graphe, $\mathbf{h}_i$ la valeur des fonctions $\mathbf{h}$ au noeud i, $\mathbf{e}_{ij}$ la valeur de la fonction sur l'arête (i, j) et $\varphi_{\boldsymbol{\theta}} : \mathbb{R}^{2in+e} \to \mathbb{R}^m$ une fonction paramétrée par un réseau de neurones type MLP ou couche linéaire.

- Exemple **GraphSage**
  - nonlocalité:

    $$\varphi_{\boldsymbol{\theta}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij}) = W\mathbf{h}_j$$

  - localité: $\gamma_\theta(\mathbf{h}, \mathbf{m}) = \sigma(\mathbf{h})$

- Local coordinates bases methods:

  $$\varphi_{\boldsymbol{\theta}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij}) = \sum_j^q \alpha_j \varphi_{\boldsymbol{\theta}_j}(u(\mathbf{x}_i, \mathbf{x}_j))\mathbf{h}_j$$

- localité: $\gamma_\theta(\mathbf{h}, \mathbf{m}) = \sigma(\mathbf{h})$

GNN

- Pour plusieurs GNN on utilise:

$$u(x, y) = (\rho(x, y), \theta(x, y))$$

  - Geodesic GNN/GNO:

$$\varphi(\theta) = e^{-\left(\begin{pmatrix}\rho\\ \theta\end{pmatrix}-\mu\right)^T \begin{pmatrix}\sigma_\rho & 0\\ 0 & \sigma_\theta\end{pmatrix}\left(\begin{pmatrix}\rho\\ \theta\end{pmatrix}-\mu\right)}$$

  - Anistropic GNN/GNO:

$$\varphi(\theta) = e^{-\left(\begin{pmatrix}\rho\\ \theta\end{pmatrix}^T R_{\theta_j}\begin{pmatrix}\alpha & 0\\ 0 & 1\end{pmatrix}R^t_{\theta_j}\begin{pmatrix}\rho\\ \theta\end{pmatrix}\right)}$$

  - Monet GNN/GNO:

$$\varphi(\theta) = \sum_{j=1}^{q} \alpha_j e^{-\left(\begin{pmatrix}\rho\\ \theta\end{pmatrix}-\mu\right)^T \Sigma^{-1}\left(\begin{pmatrix}\rho\\ \theta\end{pmatrix}-\mu\right)}$$

avec Σ, μ des paramètres apprenables.