SciML 2: Generative models

E. Franck

Master CSMI, Strasbourg University

01/09/2025

1. Principe et 1eres approches

Types d'Apprentissage

- Pour le moment on a considérer l'apprentissage supervisé:
  - On a des données d'entrées et de sorties
  - On cherche à apprendre une relation déterministe qui mappe les entrées aux sorties

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \mathcal{L}(f(x_i; \boldsymbol{\theta}), y_i)$$

- Autre type d'apprentissage:
  - Apprentissage non supervisé (déterminer des structures dans les données sans sorties associées)
  - Apprentissage par renforcement (déterminer une politique d'actions pour maximiser une récompense cumulative)
- **Apprentissage non supervisé**:
  - clustering,
  - réduction de dimensionnalité,
  - dictionnaires creux,
  - estimation de densité et modèles génératifs, etc.

Modèles génératifs

Objectives (Estimation de densité): Soit des données $\mathbb{x}_1, ..., \mathbb{x}_n \in \mathbb{R}^d$ issues d'une distribution inconnue de desnité $p_{target}(\mathbf{x})$. On souhaite estimer cette distribution, c'est-à-dire construire un modèle $p_{\boldsymbol{\theta}}(\mathbf{x})$ qui approxime $p_{target}(\mathbf{x})$.

Objectives (Modèles génératifs): Soit des données $\mathbb{x}_1, ..., \mathbb{x}_n \in \mathbb{R}^d$ issues d'une distribution inconnue de densité $p_{target}(\mathbf{x})$. Un modèle génératif $q_{\boldsymbol{\theta}}$ est processus capable de générer de nouvelles données $\tilde{\mathbb{x}}$) selon une loi qui approche $p_{target}(\mathbf{x})$.

- Un modèle d'estimation de densité est aussi un modèle génératif.
- Un modèle génératif ne donne pas forcément une estimation de densité explicite. Il s'agit d'un processus d'echantillonnage. Il caractérise donc implicitement la densité.

Objectives (modèles génératif conditionnels): On peut aussi vouloir construire des modèles génératifs conditionnels. C'est-à-dire des modèles capables de générer des données $\mathbf{x}$ conditionnellement à une entrée $\mathbf{y}$. On caractérise donc implicitement la densité conditionnelle $p_{target}(\mathbf{x} \mid \mathbf{y})$.

Applications

- Génération d'images réalistes (GANs, VAEs)
- Synthèse de texte (modèles de langage, GPT)
- Modélisation de la distribution de données complexes (flux normaux, modèles variationnels)
- Synthèse de molécules en chimie et biologie computationnelle
- Simulation de systèmes physiques (modèles basés sur la physique, réseaux de neurones génératifs)
- Amélioration de la résolution d'images (super-résolution)
- Traduction automatique et génération de discours

Qu'est ce qu'on minimise ?

- **Maximum de vraisemblance**: qui consiste à maximiser la probabilité des données observées sous le modèle.

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(\mathbb{x}_i)$$

- **Minimisation de la divergence entre la distribution réelle et la distribution modélisée**:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D\big(p_{\text{target}} \parallel p_{\boldsymbol{\theta}}\big) = \arg\min_{\boldsymbol{\theta}} \int p_{\text{target}}(\mathbf{x}) \log\left(\frac{p_{\text{target}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\right) d\mathbf{x}$$

- Les deux approches sont équivalentes lorsque la divergence choisie est la divergence de Kullback-Leibler (KL).
- Pk ?
  - En pratique on a souvent pas accès à p_{target} mais seulement à des échantillons issus de cette distribution.
  - Kl vers Maximum de vraisemblance:
  - On approxime l'intégrale par une somme sur les échantillons.

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \int p_{\text{target}}(\mathbf{x}) \log\left(\frac{p_{\text{target}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\right) d\mathbf{x} \approx \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log\left(\frac{p_{\text{target}}(\mathbb{x}_i)}{p_{\boldsymbol{\theta}}(\mathbb{x}_i)}\right) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log(p_{\boldsymbol{\theta}}(\mathbb{x}_i))$$

1er modèle de densité: mélange de Gaussiennes

- On modélise la densité comme une combinaison linéaire de K densités gaussiennes:

$$p_\theta(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

où $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{\{k=1\}}^{K}$ sont les paramètres du modèle, avec π_k les poids des composantes (somme à 1), μ_k les moyennes et Σ_k les matrices de covariance.

- On peut apprendre les paramètres via l'algorithme EM (Expectation-Maximization)
- Implicitement cet algorithm maximise la vraisemblance des données sous le modèle.

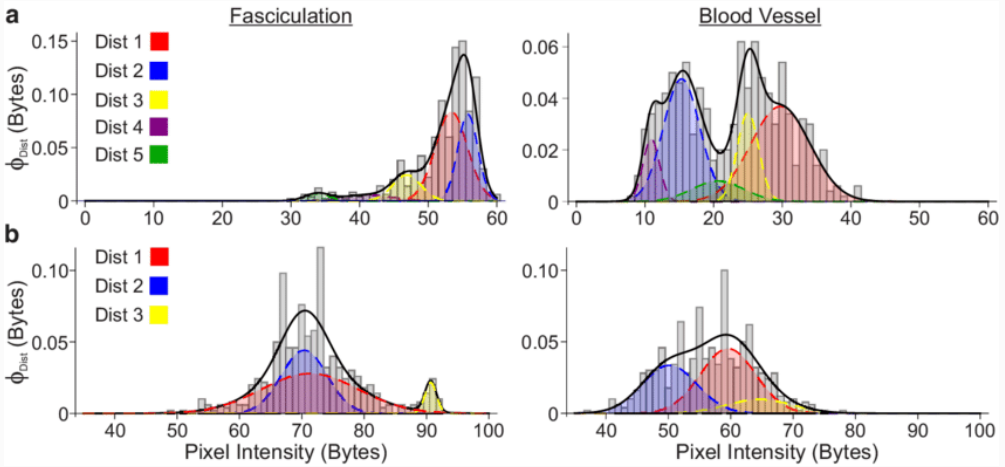


Figure 1: "Exemple de mélange de Gaussiennes pour modéliser une distribution complexe. Google Image"

1er modèle génératif: auto-encodeur variationnel (VAE)

- Un VAE est un modèle génératif probabiliste qui apprend une représentation latente des données.
- Il se compose de deux parties:
  - Un encodeur qui mappe les données d'entrée $\mathbb{x}$ vers une distribution latente $q_{\varphi}(\mathbf{z} \mid \mathbb{x})$.
  - Un décodeur qui mappe les échantillons de la distribution latente vers la distribution des données $p_{\theta}(\mathbb{x} \mid \mathbf{z})$.

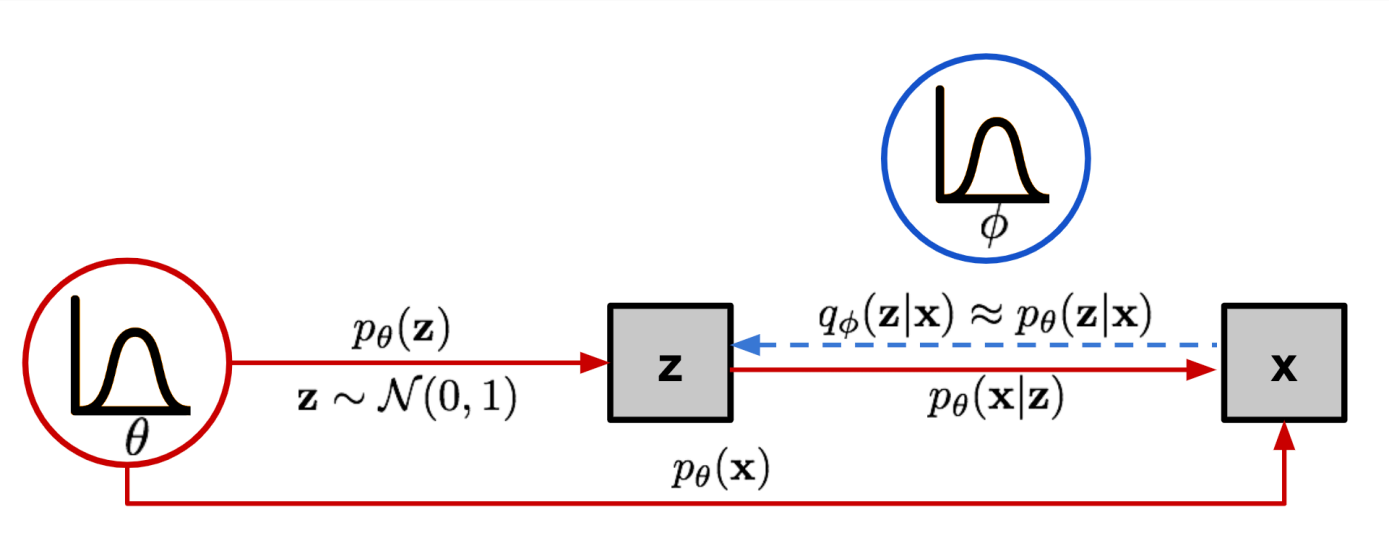


Figure 2: VAE. Lil Blog

- L'apprentissage consiste à maximiser la vraissemblance marginale des données. En pratique on utilise une estimation inférieure (ELBO).

2. Flot Normalisé

Idée principale

- Contrairement a d'autres méthodes les flots normalisés permettent de construire explicitement la densité.

Idea : Soit la densité cible $p_{target}(\mathbf{x})$. L'idée est de construire une transformation bijective $f : \mathbb{R}^d \to \mathbb{R}^d$ telle que si $\mathbf{z} \sim p_0(\mathbf{x})$ connue alors $\mathbf{x} = f(\mathbf{z})$ suit la densité cible $p_{target}(\mathbf{x})$.

- Par définition des loi de probabilité:

$$\int_X p_{target}(\mathbf{x})d\mathbf{x} = \int_Z p_0(\mathbf{z})d\mathbf{z} = \int_{f^{-1}(X)} p_0(\mathbf{z})d\mathbf{z} = 1$$

- Par le changement de variable $\mathbf{z} = f^{-1}(\mathbf{x})$ on a:

$$\int_X p_{target}(\mathbf{x})d\mathbf{x} = \int_X p_0(f^{-1}(\mathbf{x}))\ |\det J_{f^{-1}}(\mathbf{x})|\ d\mathbf{x} = 1$$

- On en déduit l'expression de la densité cible:

$$p_{target}(\mathbf{x}) = p_0(f^{-1}(\mathbf{x}))\ |\det J_{f^{-1}}(\mathbf{x})|$$

Principe 2

Remark: Si on a une transformation bijective f alors on peut donc calculer p_{target} à partir de p_0 et de f. Encore plus facilement on échantillonner de p_{target} en échantillonnant de p_0 et en appliquant f.

Idea: Une composition de transformations bijectives est aussi une transformation bijective. On peut donc construire des transformations complexes en composant des transformations simples.

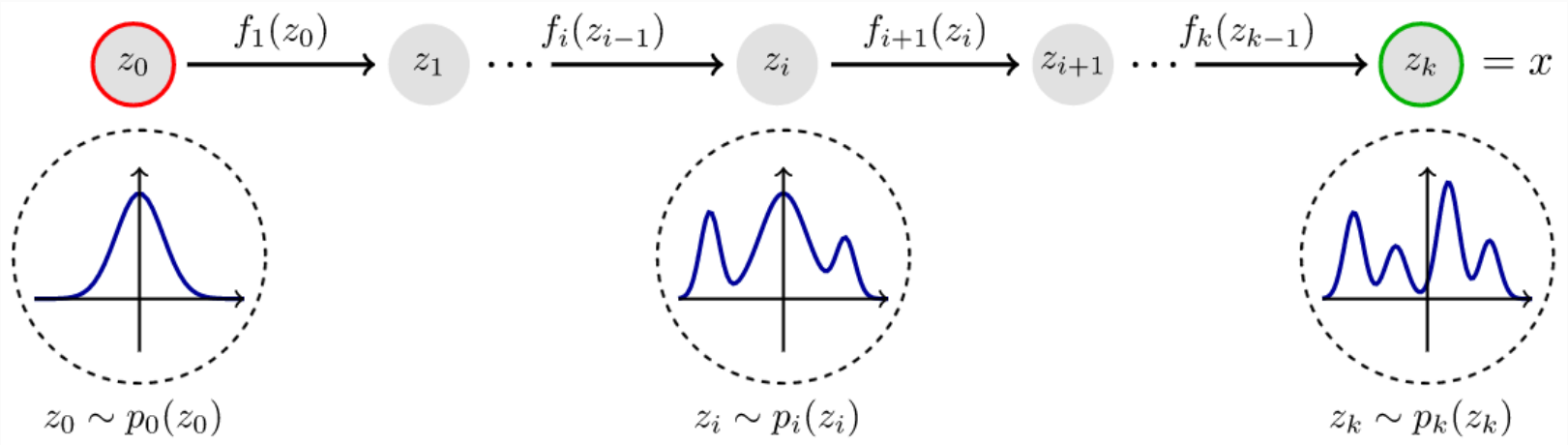


Figure 3: Flow normalisé. Lil Blog

Flot normalisé

Definition (Flots normalisés): Soit $p_0(\mathbf{z})$ une densité de probabilité simple. Un flot normalisé est une transformation bijective $f = f_L \circ \ldots \circ f_1$ telle qu'un échantillon de la loi final $p_{\boldsymbol{\theta}}(\mathbf{x})$ soit obtenue en échantillonnant $\mathbf{z} \sim p_0(\mathbf{z})$ puis en calculant $\mathbf{x} = f(\mathbf{z})$.

- On peut calculer la densité finale via la formule du changement de variable:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = p_0(f^{-1}(\mathbf{x})) \prod_{i=1}^{L} |\det J_{f_i^{-1}}(\mathbf{h}_i)|$$

où $\mathbf{h}_i = f_{i-1} \circ \ldots \circ f_1(\mathbf{z})$ et $\mathbf{h}_0 = \mathbf{z}$.

Proposition (log probabilité): La log probabilité de $\mathbf{x}$ sous le modèle est donnée par:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log p_0(f^{-1}(\mathbf{x})) + \sum_{i=1}^{L} \log |\det J_{f_i^{-1}}(\mathbf{h}_i)|$$

Apprentissage

- L'objectif est de minimiser la divergence entre la densité cible $p_{\text{target}}(\mathbf{x})$ et la densité modélisée $p_{\boldsymbol{\theta}}(\mathbf{x})$.

Proposition: Minimiser la divergence de KL revient à maximiser la log vraisemblance des données sous le modèle:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D_{\text{KL}}\left(p_{\text{target}} \parallel p_{\boldsymbol{\theta}}\right) = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbb{x}_i)$$

- La formule de la log probabilité permet de calculer facilement le gradient pour l'optimisation.
- On peut utiliser des algorithmes d'optimisation stochastique (SGD, Adam, etc.) pour apprendre les paramètres $\boldsymbol{\theta}$ des transformations f_i
- Comment choisir les transformations f_i ?

Remark (Besoin):
- transformations bijectives
- inversibles facilement calculables
- jacobiennes avec déterminant calculable efficacement

Analogie avec les EDO

- Soit $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^d$. Une façon de construire une transformation bijective entre $\mathbf{x}_0$ et $\mathbf{x}_1$ est de considérer l'EDO:

$$\begin{cases} \frac{d\mathbf{x}(t)}{dt} = \mathbf{v}(\mathbf{x}(t)) \\ \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{x}(1) = \mathbf{x}_1 \end{cases}$$

- En effet par le théorème de Cauchy-Lipschitz, si $\mathbf{v}$ est suffisamment régulier le flot de cette EDO est reversible en en temps: $\Phi_t(0, t, \mathbf{x}_0) = \Phi_{-t}(\Phi_t(\mathbf{x}_0)) = I_d$

Idea:
- on paramétrise des flux d'EDO
- On discrétise l'EDO pour obtenir une couche
- Si on utilise schéma reversible ou calcul ou si peut calculer le flot exact on aura une couche **inversible**.

Exemple: Nice

- On se donne l'EDO

$$\frac{\mathrm{d}\mathbf{x}(\mathrm{t})}{\mathrm{dt}} = \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(\mathrm{t})),$$

- On split $\mathbf{x}$ en deux parties $\mathbf{x} = (\mathbf{x}_\mathrm{a}, \mathbf{x}_\mathrm{b})$ et obtient les deux sous équations

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}\mathbf{v}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x})\\ 0\end{pmatrix}, \qquad \frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}0\\ \mathbf{v}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x})\end{pmatrix}$$

- On choisit la paramétrisation:

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}\mathbf{v}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\\ 0\end{pmatrix}, \qquad \frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}0\\ \mathbf{v}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a})\end{pmatrix}$$

- On applique le schéma d'Euler explicite pour obtenir les transformations:

$$\mathrm{f}_1 : \begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{x}_\mathrm{a} + \Delta\mathrm{t}\mathbf{v}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\\ \mathbf{x}_\mathrm{b}\end{pmatrix}, \quad \mathrm{f}_2 : \begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b} + \Delta\mathrm{t}\mathbf{v}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a})\end{pmatrix}$$

- Ses flot discrets sont exacts pour chaque sous système donc reversible en temps et donc inversible.

Exemple: Nice II

Definition (Couche Nice): Une couche Nice est une transformation bijective $\mathrm{f} = \mathrm{f}_2 \circ \mathrm{f}_1$ où f_1 et f_2 sont définies par:

$$\mathrm{f}_1 : \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{x}_\mathrm{a} + \mathbf{v}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b}) \\ \mathbf{x}_\mathrm{b} \end{pmatrix}, \quad \mathrm{f}_2 : \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} + \mathbf{v}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a}) \end{pmatrix}$$

avec $\mathrm{v}_{\boldsymbol{\theta},\mathrm{a}}$ et $\mathrm{v}_{\boldsymbol{\theta},\mathrm{b}}$ des réseaux de neurones.

- Le déterminant de la jacobienne est facile à calculer: $\det \mathrm{J}_\mathrm{f}(\mathbf{x}) = 1$
- L'inverse est aussi facile à calculer:

$$\mathrm{f}_1^{-1} : \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{x}_\mathrm{a} - \mathbf{v}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b}) \\ \mathbf{x}_\mathrm{b} \end{pmatrix}, \quad \mathrm{f}_2^{-1} : \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} \end{pmatrix} \mapsto \begin{pmatrix} \mathbf{x}_\mathrm{a} \\ \mathbf{x}_\mathrm{b} - \mathbf{v}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a}) \end{pmatrix}$$

- Globalement en cherchant des splitting ou des EDO nilpotentes on peut construire des transformations bijectives avec des jacobiennes à déterminants calculables efficacement.

Exemple: RealNVP

- $$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b}) \odot \mathbf{x}_\mathrm{a} + \mathbf{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\\ 0\end{pmatrix}, \qquad \frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}0\\ \mathbf{w}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a}) \odot \mathbf{x}_\mathrm{b} + \mathbf{s}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a})\end{pmatrix}$$

  On va considère que la 1er couche mais la situation est similaire pour la seconde couche. On a

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} = \begin{pmatrix}\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b}) \odot \mathbf{x}_\mathrm{a} + \mathbf{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\\ 0\end{pmatrix}$$

-On résout l'équation sur $\mathbf{x}_\mathrm{a}$ en considérant $\mathbf{x}_\mathrm{b}$ comme une constante cela donne

$$\mathbf{x}_\mathrm{a}(\mathrm{t}) = \exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\big)\mathrm{t}\big) \odot \mathbf{x}_\mathrm{a}(0) + \int_0^{\Delta\mathrm{t}} \exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\big)\tau\big) \odot \mathbf{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\mathrm{d}\tau$$

- On utilise le fait que si $\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})$ est inversible on a

$$\int_0^{\Delta\mathrm{t}} \exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\big)\tau\big)\mathrm{d}\tau = \mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\big)^{-1} \odot \big(\exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\big)\Delta\mathrm{t}\big) - \mathbf{1}\big)$$

Exemple: RealNVP

- ce qui donne

$$\mathbf{x}_{\mathrm{a}}(\mathrm{t}) = \exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big)\mathrm{t}\big) \odot \mathbf{x}_{\mathrm{a}}(0) + \Big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big)^{-1} \odot \big(\exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big)\mathrm{t}\big) - \mathbf{1}\big)\Big) \odot \mathbf{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})$$

- Vu que dans l'idée $\mathbf{s}$ et $\mathbf{w}$ sont des réseaux de neurones on pose $\mathbf{t}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}}) = \Big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big)^{-1} \odot \big(\exp\big(\mathrm{diag}\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big)\mathrm{t}\big) - \mathbf{1}\big)\Big) \odot \mathbf{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})$. En faisant ça sur les deux équations on obtient les transformations:

$$\mathrm{f}_1 : \begin{pmatrix}\mathbf{x}_{\mathrm{a}}\\ \mathbf{x}_{\mathrm{b}}\end{pmatrix} \mapsto \begin{pmatrix}\exp\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big) \odot \mathbf{x}_{\mathrm{a}} + \mathbf{t}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\\ \mathbf{x}_{\mathrm{b}}\end{pmatrix}, \quad \mathrm{f}_2 : \begin{pmatrix}\mathbf{x}_{\mathrm{a}}\\ \mathbf{x}_{\mathrm{b}}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{x}_{\mathrm{a}}\\ \exp\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_{\mathrm{a}})\big) \odot \mathbf{x}_{\mathrm{b}} + \mathbf{t}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_{\mathrm{a}})\end{pmatrix}$$

qui sont obtenues en prenant le flot exact d'une EDO ce qui assure l'inversibilité.

Definition (Couche RealNVP): Une couche RealNVP est une transformation bijective $\mathrm{f} = \mathrm{f}_2 \circ \mathrm{f}_1$ où f_1 et f_2 sont définies par:

$$\mathrm{f}_1 : \begin{pmatrix}\mathbf{x}_{\mathrm{a}}\\ \mathbf{x}_{\mathrm{b}}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{x}_{\mathrm{a}} \odot \exp\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\big) + \mathbf{t}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_{\mathrm{b}})\\ \mathbf{x}_{\mathrm{b}}\end{pmatrix}, \quad \mathrm{f}_2 : \begin{pmatrix}\mathbf{x}_{\mathrm{a}}\\ \mathbf{x}_{\mathrm{b}}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{x}_{\mathrm{a}}\\ \mathbf{x}_{\mathrm{b}} \odot \exp\big(\mathbf{w}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_{\mathrm{a}})\big) + \mathbf{t}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_{\mathrm{a}})\end{pmatrix}$$

avec $\mathbf{w}_{\boldsymbol{\theta},\mathrm{a}}$, $\mathbf{w}_{\boldsymbol{\theta},\mathrm{b}}$, $\mathbf{t}_{\boldsymbol{\theta},\mathrm{a}}$ et $\mathbf{t}_{\boldsymbol{\theta},\mathrm{b}}$ des réseaux de neurones.

Exemple: RealNVP III

- Le déterminant de la jacobienne est facile à calculer: $\det \mathrm{J}_{\mathrm{f}_1}(\mathbf{x}) = \exp\left(\sum\left(\mathrm{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\right)\right)$
- L'inverse est aussi facile à calculer:

$$\mathrm{f}_1^{-1} : \begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} \mapsto \begin{pmatrix}\left(\mathbf{x}_\mathrm{a} - \mathrm{t}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\right) \odot \exp\left(-\mathrm{s}_{\boldsymbol{\theta},\mathrm{a}}(\mathbf{x}_\mathrm{b})\right)\\ \mathbf{x}_\mathrm{b}\end{pmatrix}, \quad \mathrm{f}_2^{-1} : \begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \mathbf{x}_\mathrm{b}\end{pmatrix} \mapsto \begin{pmatrix}\mathbf{x}_\mathrm{a}\\ \left(\mathbf{x}_\mathrm{b} - \mathrm{t}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a})\right) \odot \exp\left(-\mathrm{s}_{\boldsymbol{\theta},\mathrm{b}}(\mathbf{x}_\mathrm{a})\right)\end{pmatrix}$$

- Example:

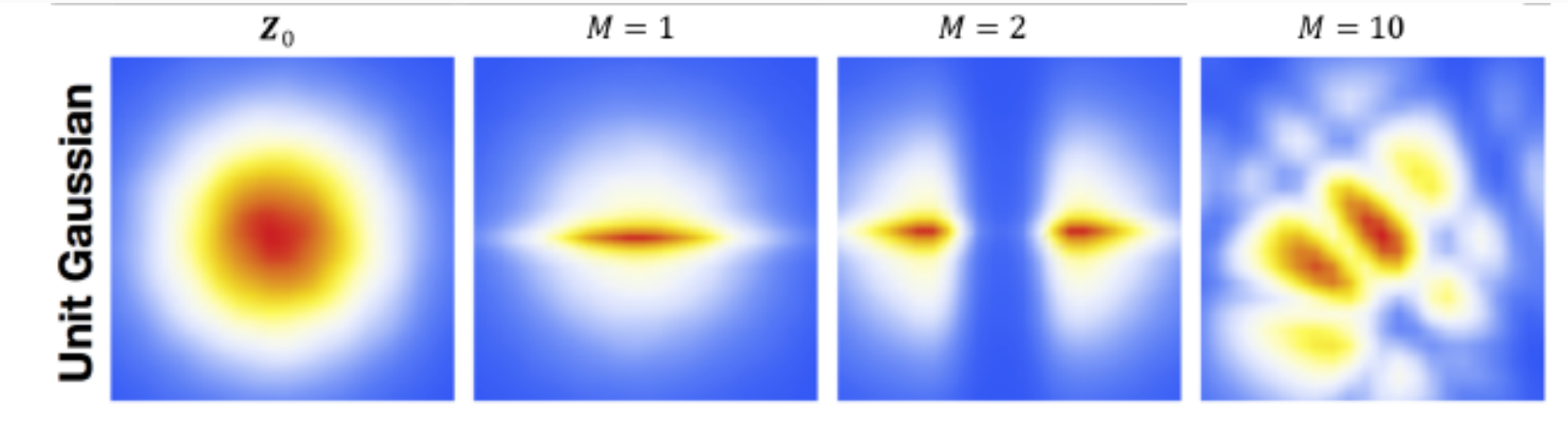


Figure 4: RealNVP. Lil Blog

Exemple: KRNet I

- idée: utiliser une transformation triangulaire:

$$\mathbf{y} = f(\mathbf{x}) = \begin{pmatrix} f_1(x_1) \\ f_2(x_1, x_2) \\ ... \\ f_d(x_1, x_2, ..., x_d) \end{pmatrix}$$

- Le réseau va donc enchaine des transformation qui vont utiliser de plus en plus de variables.

$$f(\mathbf{x}) = f_N \circ f_{O,d}(\mathbf{x}) \circ f_{O,d-1}(\mathbf{x}) \circ ... \circ f_{O,1}(\mathbf{x})$$

- L'enjeu est maintenant de construire des transformations $f_{O,i}$ bijectives avec Jacobienne à déterminant calculable efficacement.
- La transformation $f_{O,k}$ ne modifie que les composantes 1 à k

Definition (Couche de Squeezing): Une couche de Squeezing est donné par $S_k(\mathbf{x}) = \mathbf{q} \odot \mathbf{x}$ avec $\mathbf{q} = \left(\underbrace{(1, ..., 1)}_{k}, \underbrace{(0, ..., 0)}_{d-k} \right)$ est une transformation bijective.

Exemple: KRNet II

Definition (Couche de rotation): Une couche de rotation est une transformation bijective $R_{\boldsymbol{\theta}}$ définie par:

$$R_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{pmatrix} L & \mathbf{0} \\ \mathbf{0} & I_d \end{pmatrix} \begin{pmatrix} U & \mathbf{0} \\ \mathbf{0} & I_d \end{pmatrix} \mathbf{x}$$

avec L, U entrainable

Definition (Block interne): Soit $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ in $\left(\mathbb{R}^k, \mathbb{R}^{d-k}\right)$ un bloc interne $f_i(\mathbf{x})$ est la compose d'une couche d'échelle donnée par:

$$f_{sca}(\mathbf{x}) = \mathbf{a} \odot \mathbf{x} + \mathbf{b}$$

avec $\mathbf{a}$ et $\mathbf{b}$ apprennable et d'une couche affine

$$f_{aff} = \begin{pmatrix} \mathbf{x}_a \\ \mathbf{x}_b \odot (1 + \alpha \tanh(t_{\boldsymbol{\theta}}(\mathbf{x}_a)) + e^{\beta} \odot \tanh(t_{\boldsymbol{\theta}}(\mathbf{x}_a))) \end{pmatrix}$$

avec α un paramètre β entrainable et $t_{\boldsymbol{\theta}}$ un réseau

Exemple: KRNet III

Definition (Block externe): Soit $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$ in $\left(\mathbb{R}^k, \mathbb{R}^{d-k}\right)$ un bloc externe

$$f_{O,k}(\mathbf{x}) = S_k \circ f_{i,L} \ldots \circ f_{i,1} \circ R_{\boldsymbol{\theta}}$$

- Il reste a définir la couche final L_N qui introduit une nonlinéarité assez particulière, appliqué composante par composante donné par :

$$f_{N(x)} = \begin{cases} \beta(x-a)+a, & x \in (-\infty, -a) \\ \varphi^{-1} \circ F \circ \varphi & \\ \beta(x+a)-a, & x \in (a, \infty) \end{cases}$$

avec φ affine et

$$F(x) = \int_0^x f(y)dy$$

Exemple: Residual approach I

- On se donne l'EDO

$$\frac{\mathrm{d}\mathbf{x}(\mathrm{t})}{\mathrm{dt}} = \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(\mathrm{t})),$$

- On approxime le flot de cette EDO via un schéma de type Euler explicite:

$$\mathbf{x}_{\mathrm{i}+1} = \mathbf{x}_{\mathrm{i}} + \Delta \mathrm{t} \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{i}})$$

- Si on veut inverser on résout $\mathrm{g}(\mathbf{x}_{\mathrm{i}}) = \mathbf{x}_{\mathrm{i}}$ par une méthode de **Point fixe** avec

$$\mathrm{g}(\mathbf{x}_{\mathrm{i}}) = \mathbf{x}_{\mathrm{i}+1} - \Delta \mathrm{t} \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{i}})$$

- Pour converger on a besoin que $\Delta \mathrm{t}$ soit suffisamment petit et que $\mathbf{v}_{\boldsymbol{\theta}}$ soit Lipschitzien avec constante de Lipschitz L telle que $\Delta \mathrm{t} \mathrm{L} < 1$.

Definition (Couche residuelle): Une couche résiduelle est une transformation bijective f définie par:

$$\mathrm{f} : \mathbf{x} \mapsto \mathbf{x} + \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x})$$

où $\mathbf{v}_{\boldsymbol{\theta}}$ est un réseau de neurones **contractant**

Exemple: Residual approach II

- Comment obtenir une paramétrisation contractante ?

Definition (Spectral normalized layer): Une couche à normalisation spectrale est une couche linéaire $\mathbf{w_\theta}$: $\mathbb{R}^d \to \mathbb{R}^d$ de la form

$$\mathbf{x}_{l+1} = W_{s,\boldsymbol{\theta}} \mathbf{x}_l$$

avec $W_{s,\boldsymbol{\theta}} = \frac{W}{\Sigma(W)}$ ou $\Sigma(W)$ est la plus grande valeur singulière de W et W la matrice de poids associée à la couche.

- La valeur singulière maximale est approximée via une version différentiable de méthode de la puissance.
- Ici on contraint chaque couche a être contractante avec constante de Lipschitz $L = 1$.

Exemple: Neural Splins flows I

Idea : On applique composante par composante une fonction monotone, bijective et différentiable construite comme une spline rationnelle quadratique.

- On se donne $[-c, c]$
- On se donne des noeuds $b_1 < b_2 < \ldots < b_{K-1}$ dans $[-c, c]$ avec $b_0 = -c$ et $b_K = c$
- On se donne des images $y_1 < y_2 < \ldots < y_{K-1}$ dans $[-c, c]$ avec $y_0 = -c$ et $y_K = c$
- On se donne des dérivées $d_0, d_1, \ldots, d_K$ en chaque noeud
- On définit la fonction spline par morceaux:

$$S(x) = \frac{(y_{i+1} - y_i)(x - b_i)^2 + d_i(x - b_i)(b_{i+1} - x)}{(b_{i+1} - b_i)^2 + \left(d_{i+1} + d_i - 2\frac{y_{i+1} - y_i}{b_{i+1} - b_i}\right)(x - b_i)(b_{i+1} - x)}$$

- Determinant jacobienne facile à calculer:
- Inverse facile à calculer via résolution d'une équation quadratique.

Exemple: Neural Splins flows II

Soit une spline vectorielle

$$\mathbf{S} = \begin{pmatrix} \mathrm{S}_1(\mathrm{x}_1) \\ \mathrm{S}_2(\mathrm{x}_2) \\ . \\ \mathrm{S}_{\frac{\mathrm{d}}{2}}\left(\mathrm{x}_{\frac{\mathrm{d}}{2}}\right) \end{pmatrix}$$

Definition (Couche Neural Spline Flow): Une couche Neural Spline Flow est une transformation bijective f définie par une spline vectorielle:

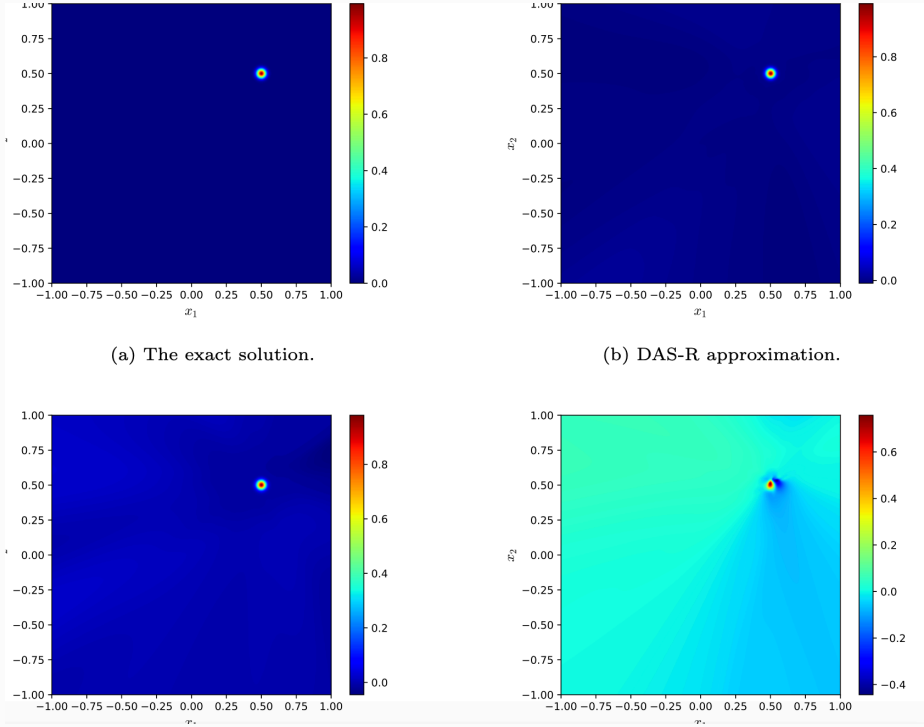
$$\mathbf{x} = \mathbf{S}(\mathbf{x}; (\mathbf{b}, \mathbf{y}, \mathbf{d})_{\mathrm{b}})$$

ou les paramètres de toutes les splines sont données par $(\mathbf{b}, \mathbf{y}, \mathbf{d})_{\mathrm{b}} = \mathrm{MLP}_{\boldsymbol{\theta}}(\mathbf{x}_{\mathrm{a}})$ et $(\mathbf{b}, \mathbf{y}, \mathbf{d})_{\mathrm{a}}$ entrainables.

Applications aux PINNs

- Dans la version primaire des PINNs: on échantillonne uniformement les PINNs.

Remark : Pour des problèmes fortement localisé ca necessite beaucoup de points. En pratique on aimerai échantillonner la ou il faut.



(a) The exact solution.

(b) DAS-R approximation.

- On veut échantillonner plus de point la ou:
  - le résidu $\mid \mathrm{r}(\mathbf{u_x}) \mid^2$ est grand
  - le gradient $\mid \nabla_{\mathbf{x}} \mathbf{u_x} \mid^2$ est grand
  - un des critères précédent sur une autre variable
- On nomme ce crtière $\mathrm{h}(\mathbf{u_\theta})$
- $\mathrm{h}(\mathbf{u_\theta})$ n'est pas une distribution de probabilité ? comment échantillonner ?

Applications aux PINNs

Idea : apprendre a fur et a mesure de l'entrainement une densité $\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}(\mathbf{x})$ qui satisfait

$$\boldsymbol{\theta}_\mathrm{p}^* = \arg\min_{\boldsymbol{\theta}_\mathrm{p}} \mathrm{KL}\Big(\mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}) \,||\, \mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}\Big)$$

- Ici $\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}} = \mathbb{1}_\Omega \circ \mathrm{FN}_{\boldsymbol{\theta}_\mathrm{p}}$ avec $\mathrm{FN}_{\boldsymbol{\theta}_\mathrm{p}}$ un flot normalisé et $\mathbb{1}_\Omega$ un algorithme qui rejette les points qui ne sont pas dans le domaine.
- On développe:

$$\arg\min_{\boldsymbol{\theta}_\mathrm{p}} \mathrm{KL}\Big(\mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}})(\mathbf{x}) \,||\, \mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}(\mathbf{x})\Big) = \arg\min_{\boldsymbol{\theta}_\mathrm{p}} \int_\Omega \mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}})(\mathbf{x}) \log\left(\frac{\mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}})(\mathbf{x})}{\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}(\mathbf{x})}\right) \mathrm{d}\mathbf{x}$$

$$= \arg\min_{\boldsymbol{\theta}_\mathrm{p}} \int_\Omega \mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}) \log(\mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}))\mathrm{d}\mathbf{x} - \arg\min_{\boldsymbol{\theta}_\mathrm{p}} \int_\Omega \mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}) \log\Big(\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}\Big)\mathrm{d}\mathbf{x} = \arg\min_{\boldsymbol{\theta}_\mathrm{p}} - \int_\Omega \mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}) \log\Big(\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}\Big)\mathrm{d}\mathbf{x}$$

- On utilise le fait que

$$\int_\Omega \mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}) \log\Big(\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}\Big)\mathrm{d}\mathbf{x} = \mathbb{E}_{\mathcal{U}}\Big[\mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}}) \log\Big(\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}\Big)\Big] = \mathbb{E}_{\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}}\left[\frac{\mathrm{h}(\mathbf{u}_{\boldsymbol{\theta}})}{\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}} \log\Big(\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}\Big)\right]$$

Applications aux PINNs II

- On applique la méthode de Monte Carlo et donc on va minimiser

$$J_p(\theta_p) = -\frac{1}{n}\sum_{i=1}^{n} \frac{h(u_\theta)(x_i)}{p_{\theta_p}(x_i)} \log\left(p_{\theta_p}(x_i)\right)$$

avec des points échantillonner par p_{θ_p}.

- Coté PINNs normalement on minimise

$$J(\theta) = \int_\Omega r(u_\theta)(x)^2 dx = \mathbb{E}_{\mathcal{U}}\left[r(u_\theta)^2\right] = \mathbb{E}_{p_{\theta_p}}\left[\frac{r(u_\theta)^2}{p_{\theta_p}}\right]$$

- Une fois discrétiser avec Monte Carlo on obtient le problème de Minimisation

$$\theta^* = \arg\min_\theta \frac{1}{n}\sum_{i=1}^{n} \frac{r(u_\theta)^2(x_i)}{p_{\theta_p}(x_i)}$$

Remark:

- Dans l'algorithme on réentraine regulière (toute les k itérations de gradient) le flot normalisé pour sampler de mieux en mieux
- Soit p_{θ_p} est combiné avec une loi uniforme pour assurer de garder un peu de point partout.

Applications aux PINNs III

- On peut proposer d'autre type d'optimisation.
- Par exemple:

$$\boldsymbol{\theta}^*, \mathrm{p}^*_{\boldsymbol{\theta}_\mathrm{p}} = \arg\min_{\boldsymbol{\theta}} \max_{\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}} \in \mathrm{V}} \int_\Omega \mathrm{r}(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}) \mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}(\mathbf{x}) \mathrm{d}(\mathbf{x})$$

avec $\mathrm{V} = \left\{ \mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}, \quad \int \mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}(\mathbf{x}) \mathrm{d}\mathbf{x} = 1, \mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}(\mathbf{x}) > 0 \right\}$

Remark:

- On peut montrer que résoudre ce problème fait tendre le résidu et

$$\frac{\mathrm{r}(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x})}{\int_\Omega \mathrm{r}(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}) \mathrm{d}\mathbf{x}} \to \mu_\Omega$$

avec μ_Ω la mesure de la loi uniforme. Cela pousse donc l'apprentissage a minimise le plus uniformement possible le résidu

- Pour que le théorème marche on a besoin que la constante de Lipschitz de $\mathrm{p}_{\boldsymbol{\theta}_\mathrm{p}}$ soit plus petite que 1.

Applications aux PINNs IV

On résout donc le problème modifié:

$$\boldsymbol{\theta}^*, \boldsymbol{\theta}_p^* = \arg\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\theta}_p} \int_\Omega r(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}) p_{\boldsymbol{\theta}_p}(\mathbf{x}) d(\mathbf{x}) - \beta \int | \nabla_{\mathbf{x}} p_{\boldsymbol{\theta}_p}(\mathbf{x}) |^2 \, d\mathbf{x}$$

Remark: On va alterner l'optimisation selon $\boldsymbol{\theta}$ et $\boldsymbol{\theta}_p$

- Pour $\boldsymbol{\theta}$ on va minimiser:

$$\int_\Omega r(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}) p_{\boldsymbol{\theta}_p}(\mathbf{x}) \approx \sum_{i=1}^n r(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}_i)$$

- Pour $\boldsymbol{\theta}_p$ on va maximiser

$$\int_\Omega r(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}) p_{\boldsymbol{\theta}_p}(\mathbf{x}) d(\mathbf{x}) - \beta \int | \nabla_{\mathbf{x}} p_{\boldsymbol{\theta}_p}(\mathbf{x}) |^2 \, d\mathbf{x} \approx \sum_{i=1}^n \frac{r(\mathbf{u}_{\boldsymbol{\theta}})^2(\mathbf{x}_i)}{p^k_{\boldsymbol{\theta}_p}(\mathbf{x}_i)} p_{\boldsymbol{\theta}_p}(\mathbf{x}_i) - \sum_{i=1}^n \frac{| \nabla_{\mathbf{x}} p_{\boldsymbol{\theta}_p}(\mathbf{x})|^2}{p^k_{\boldsymbol{\theta}_p}(\mathbf{x}_i)}$$

ou $p^k_{\boldsymbol{\theta}_p}(\mathbf{x}_i))$ est la distribution à l'itération d'avant utilisée pour échantilloner

3. Flot Normalisé continue

version continue

- On a vu que les flot normalisé se comprennaient comme des discrétisations reversibles d'EDO.
- On peut donc directement travailler avec les EDO.

Soit $\mathbf{x}_{\text{target}}$ un échantillon de la distribution cible $\mathrm{p}_{\text{target}}$ et $\mathbf{z}_0$ un échantillon de la distribution simple p_0.

- On considère l'EDO:

$$\begin{cases} \frac{\mathrm{d}\mathbf{x}(\mathrm{t})}{\mathrm{dt}} = \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(\mathrm{t}), \mathrm{t}) \\ \mathbf{x}(0) = \mathbf{z}_0 \end{cases}$$

Remark: Si on a une particule qui suit l'équation précédente alors la densité $\mathrm{p}(\mathbf{x}, \mathrm{t})$ de la particule à l'instant t satisfait l'équation de continuité:

$$\frac{\partial \mathrm{p}(\mathbf{x}, \mathrm{t})}{\partial \mathrm{t}} + \nabla_{\mathbf{x}} \cdot (\mathrm{p}(\mathbf{x}, \mathrm{t}) \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, \mathrm{t})) = 0$$

- On développe

$$\frac{\partial \mathrm{p}(\mathbf{x}, \mathrm{t})}{\partial \mathrm{t}} = -\mathrm{p}(\mathbf{x}, \mathrm{t}) \nabla_{\mathbf{x}} \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, \mathrm{t}) - \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}, \mathrm{t}) \cdot \nabla_{\mathbf{x}} \mathrm{p}(\mathbf{x}, \mathrm{t})$$

Flot Normalisé continue

- Maintenant on regarde l'évolution de la densité pour une particule $\mathbf{x}(t)$:

$$\frac{d}{dt}p(\mathbf{x}(t),t) = \frac{\partial p(\mathbf{x},t)}{\partial t} + \frac{d\mathbf{x}(t)}{dt} \cdot \nabla_{\mathbf{x}} p(\mathbf{x},t)$$

- Maintenant on introduit l'équation de continuité:

$$\frac{d}{dt}p(\mathbf{x}(t),t) = -p(\mathbf{x}(t),t)\nabla_{\mathbf{x}} \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(t),t)$$

- On multiplie par $\frac{1}{p(\mathbf{x}(t),t)}$ et on intègre en temps:

$$\frac{d}{dt}\log p(\mathbf{x}(t),t) = -\nabla_{\mathbf{x}} \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(t),t)$$

Proposition: En intégrant on obtient la log densité de la particule à l'instant t:

$$\log p(\mathbf{x}(t),t) = \log p_0(\mathbf{x}(0)) - \int_0^t \nabla_{\mathbf{x}} \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(\tau),\tau)d\tau = \log p_0(\mathbf{z}_0)$$

Flot Normalisé continue

- En pratique on va donc minimiser

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \log \mathrm{p}_{t=1}\left(\mathbf{x}_{\text{target}}\right) = \arg\min_{\boldsymbol{\theta}} -\log \mathrm{p}_0(\mathbf{x}(0)) + \int_0^1 \nabla_{\mathbf{x}} \cdot \mathbf{v}_{\boldsymbol{\theta}}(\mathbf{x}(\tau), \tau) \mathrm{d}\tau$$

- Cela necessite de résoudre l'EDO avant pour obtenir $\mathbf{x}(\tau)$.
- Le calcul de la divergence de $\mathbf{v}_{\boldsymbol{\theta}}$ peut être couteux en haute dimension.
- Le calcul du gradient necessite soit de calculer l'adjoint soit de différentier par retro-propagation dans le schéma en temps.

Remark: Pad facilement utilisable en pratique

- La méthode qu'on va introduire permet de conserver l'approche ODE en simplifiant l'entrainement.

4. Flow matching

Principe

Remark: Les méthodes de flow matching utilise une edo pour transformer une loi simple en une loi cible mais propose un entrainement nettement plus simple que précédemment

Objectives: On souhaite trouver $\mathbf{v}(\mathbf{x}, t)$ tel que la densité générée p_t tel que $p_{t=0} = p_0$ la densité de base et $p_{t=1} = p_{target}$ la densité cible. Pour cela on a accès a des échantillons de p_{target} d'échantillon $\mathbf{x}_1$ et on échantillonne $\mathbf{x}_0$ de p_0.

- On va commencer par définir l'objectif d'entrainement de la méthode flow matching.

Definition (Fonction de coût de la méthode flow matching): On souhaite minimiser la fonction de coût:

$$\mathbb{E}_{\mathbf{x}(t)\sim p_t}\left[\| \mathbf{v}_{t,\boldsymbol{\theta}}(\mathbf{x}(t)) - \mathbf{v}_t^*(\mathbf{x}(t)) \|^2\right]$$

- En pratique le champ de vitesse cible $\mathbf{v}^*$ est inconnu. L'enjeu de la méthode est de remplacer cette fonctin de coût par une fonction de coût calculable.

Construction I

- L'idée est d'utilise la connaissance des échantillons de p_{target}
- On va définir la trajectoire de notre distribution à partir d'une densité conditionnelle $p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1)$

$$p_t(\mathbf{x}) = \int p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1) p_{target}(\mathbf{x}_1) d\mathbf{x}_1$$

avec $p_0(\mathbf{x} \mid \mathbf{x}_1) = p_0(\mathbf{x})$ et $p_1(\mathbf{x} \mid \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1)$

Definition (Chemin de probabilité conditionel): Le chemin de probabilité conditionel est défini par la densité conditionnelle $p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1)$ telle que $p_0(\mathbf{x} \mid \mathbf{x}_1) = p_0(\mathbf{x})$ et $p_1(\mathbf{x} \mid \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1)$

- Par la suite on va
  - exprimer le champ de vitesse cible $\mathbf{v}^*$ en fonction d'un champ conditionnel et de $p_{t \mid 1}$
  - Montrer que le champ généré à partir du champ conditionnel est le même que celui généré par le champ de vitesse cible
  - Monter que la fonction de coût peut s'écrire en fonction du champ conditionnel uniquement.
  - proposer des chemin conditionnels simples pour lesquels on peut calculer le champ conditionnel explicitement.
  - Entraîner le modèle via la fonction de coût simplifiée.

Construction II

- On suppose que la densité conditionnelle $p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1)$ satisfait l'équation de continuité:

$$\frac{\partial p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1)}{\partial t} + \nabla_{\mathbf{x}} \cdot \left(p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1) \mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) \right) = 0$$

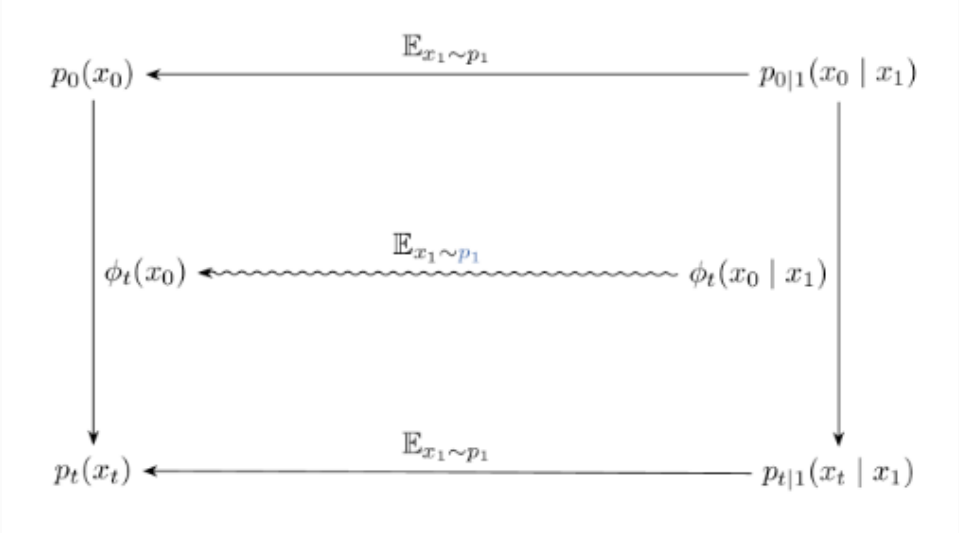


Figure 6: Schéma conditionnel. Cours flow matching Cambridge

- On peut aussi definir le champ de vitesse à partir du champ conditionnel:

$$\mathbf{v}_t(\mathbf{x}) = \mathbb{E}_{p_{1 \mid t}(\mathbf{x}_1 \mid \mathbf{x})}[\mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1)] = \int \mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) p_{1 \mid t}(\mathbf{x}_1 \mid \mathbf{x}) d\mathbf{x}_1$$

Proposition: Le champ de vitesse $\mathbf{v}_t(\mathbf{x})$ défini ci-dessus génère la même trajectoire de densité p_t que le champ de vitesse cible $\mathbf{v}_t^*(\mathbf{x})$.

Construction III

$$
\begin{aligned}
\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{p_t} &= \int \frac{\partial \mathrm{p_{t\mid 1}}(\mathbf{x}\mid \mathbf{x}_1)}{\partial \mathrm{t}}\mathrm{p_{target}}(\mathbf{x}_1)\mathrm{d}\mathbf{x}_1 \\
&= -\int \nabla_{\mathbf{x}} \cdot \Big(\mathrm{p_{t\mid 1}}(\mathbf{x}\mid \mathbf{x}_1)\mathbf{v_t}(\mathbf{x}\mid \mathbf{x}_1)\Big)\mathrm{p_{target}}(\mathbf{x}_1)\mathrm{d}\mathbf{x}_1, \\
&= -\int \nabla_{\mathbf{x}} \cdot \Big(\mathrm{p_{t\mid 1}}(\mathbf{x}\mid \mathbf{x}_1)\mathbf{v_t}(\mathbf{x}\mid \mathbf{x}_1)\mathrm{p_{target}}(\mathbf{x}_1)\Big)\mathrm{d}\mathbf{x}_1 \\
&= -\nabla_{\mathbf{x}} \cdot \left(\int \mathrm{p_{t\mid 1}}(\mathbf{x}\mid \mathbf{x}_1)\mathbf{v_t}(\mathbf{x}\mid \mathbf{x}_1)\mathrm{p_{target}}(\mathbf{x}_1)\mathrm{d}\mathbf{x}_1\right) \\
&= -\nabla_{\mathbf{x}} \cdot \left(\int \mathbf{v_t}(\mathbf{x}\mid \mathbf{x}_1)\frac{\mathrm{p_{t\mid 1}}(\mathbf{x}\mid \mathbf{x}_1)\mathrm{p_{target}}(\mathbf{x}_1)}{\mathrm{p_t}(\mathbf{x})}\mathrm{p_t}(\mathbf{x})\mathrm{d}\mathbf{x}_1\right) \\
&= -\nabla_{\mathbf{x}} \cdot \left(\int \mathbf{v_t}(\mathbf{x}\mid \mathbf{x}_1)\mathrm{p_{1\mid t}}(\mathbf{x}_1\mid \mathbf{x})\mathrm{p_t}(\mathbf{x})\mathrm{d}\mathbf{x}_1\right)
\end{aligned}
$$

- par le thèorème de Bayes.Ensuite par la definition de $\mathrm{u_t}$ on a:

$$
\frac{\mathrm{d}}{\mathrm{dt}}\mathrm{p_t} = -\nabla_{\mathbf{x}} \cdot (\mathbf{v_t}(\mathbf{x})\mathrm{p_t}(\mathbf{x}))
$$

Construction IV

- Maintenant qu'on a prouvé que le champ de vitesse générée par le champ conditionnel est le même que celui généré par le champ de vitesse cible on peut donc résoudre **l 'EDO associé au champ de vitesse conditionnel et prendre l'espérance conditionnelle pour obtenir la densité à l'instant t**.

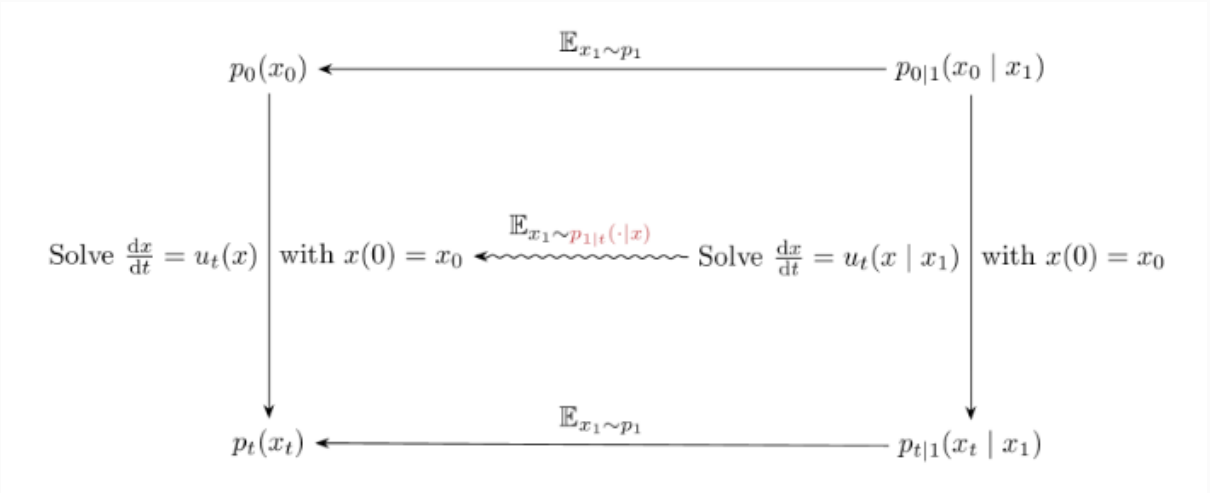


Figure 7: Schéma conditionnel. Cours flow matching Cambridge

- L'objectif de base était de minimiser

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathrm{t} \in \mathcal{U}[0,1], \mathbf{x} \sim \mathrm{p_t}} \left[\| \ \mathbf{v}_{\mathrm{t},\boldsymbol{\theta}}(\mathbf{x}_\mathrm{t}) - \mathbf{v}_\mathrm{t}^*(\mathbf{x}_\mathrm{t}) \ \|^2 \right]$$

- Puisque $\mathrm{v}(\mathbf{x}(\mathrm{t})) = \mathbb{E}_{\mathrm{p}_{1 \mid \mathrm{t}}(\mathbf{x}_1 \mid \mathbf{x})}[\mathbf{v}_\mathrm{t}(\mathbf{x} \mid \mathbf{x}_1)]$ génère la même trajectoire de densité que $\mathbf{v}_\mathrm{t}^*(\mathbf{x})$ on peut minimiser

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathrm{t} \in \mathcal{U}[0,1], \mathbf{x} \sim \mathrm{p_t}} \left[\| \ \mathbf{v}_{\mathrm{t},\boldsymbol{\theta}}(\mathbf{x}_\mathrm{t}) - \mathrm{v}(\mathbf{x}_\mathrm{t}) \ \|^2 \right]$$

Flow matching conditionnel

Theorem: Soit la fonction de coût

$$\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) = \mathbb{E}_{t\in\mathcal{U}[0,1],\mathbf{x}\sim p_t}\left[\| \mathbf{v}_{t,\boldsymbol{\theta}}(\mathbf{x}_t) - v(\mathbf{x}_t) \|^2\right]$$

et la fonction de coût

$$\mathcal{L}_{\mathrm{CM}}(\boldsymbol{\theta}) = \mathbb{E}_{t\in\mathcal{U}[0,1],\mathbf{x}_1\sim p_{\mathrm{target}},\mathbf{x}_t\sim p_{t(\mathbf{x}\mid\mathbf{x}_1)}}\left[\| \mathbf{v}_{t,\boldsymbol{\theta}}(\mathbf{x}_t) - \mathbf{v}_t(\mathbf{x}_t \mid \mathbf{x}_1) \|^2\right]$$

alors on a l'égalité

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{FM}}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathrm{CM}}(\boldsymbol{\theta})$$

- En pratique on va donc minimiser la fonction de coût conditionnelle.

Remark: La magie de la méthode c'est en se donnant un chemin conditionnel on peut calculer explicitement le champ conditionnel $\mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1)$ et donc la fonction de coût

Chemin de probabilité conditionnel

Definition (Chemin de probabilité conditionnel gaussien) : Le chemin de probabilité conditionnel gaussien est défini par la densité conditionnelle

$$p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1) = \mathcal{N}(\mathbf{x}; \mu_t(\mathbf{x}_1), \sigma^2(\mathbf{x}_1) I_d)$$

Proposition : Le champ de vitesse conditionnel associé au chemin de probabilité conditionnel gaussien est:

$$\mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) = \frac{d\mu_t(\mathbf{x}_1)}{dt} + \frac{d\sigma_t(\mathbf{x}_1)}{dt} \sigma_t(\mathbf{x}_1)^{-1} (\mathbf{x} - \mu_t(\mathbf{x}_1))$$

- Preuve:
  - Le chemin choisit implique que $x_{t \mid 1} = \mu_t(\mathbf{x}_1) + \sigma_t(\mathbf{x}_1)\mathbf{x}_0$ avec $\mathbf{x}_0 \sim p_0 = \mathcal{N}(\mathbf{0}, I_d)$
  - On sait que $\frac{\partial p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1)}{\partial t} + \nabla_{\mathbf{x}} \cdot \left(p_{t \mid 1}(\mathbf{x} \mid \mathbf{x}_1) \mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) \right) = 0$ donc

$$\frac{d\mathbf{x}_{t \mid 1}}{dt} = \mathbf{v}_t\left(\mathbf{x}_{t \mid 1} \mid \mathbf{x}_1\right)$$

Chemin de probabilité conditionnel II

- On a donc

$$\frac{\mathrm{d}\mathbf{x}_{t \mid 1}}{\mathrm{d}t} = \mathbf{v}_t\left(\mathbf{x}_{t \mid 1} \mid \mathbf{x}_1\right)$$

- Il suffit de calculer la dérivée de $\mathbf{x}_{t \mid 1}$ on obtient

$$\mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) = \frac{\mathrm{d}\mu_t(\mathbf{x}_1)}{\mathrm{d}t} + \frac{\mathrm{d}\sigma_t(\mathbf{x}_1)}{\mathrm{d}t}\mathbf{x}_0$$

- On utilise le chemin pour exprimer $\mathbf{x}_0$ en fonction de $\mathbf{x}$: $\mathbf{x}_0 = \sigma_t(\mathbf{x}_1)^{-1}(\mathbf{x} - \mu_t(\mathbf{x}_1))$.
- On obtient bien:

$$\mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) = \frac{\mathrm{d}\mu_t(\mathbf{x}_1)}{\mathrm{d}t} + \frac{\mathrm{d}\sigma_t(\mathbf{x}_1)}{\mathrm{d}t}\sigma_t(\mathbf{x}_1)^{-1}(\mathbf{x} - \mu_t(\mathbf{x}_1))$$

Remark: Un chemin classique est un chemin de probabilité conditionnel gaussien avec

$$\mu_t(\mathbf{x}_1) = t\mathbf{x}_1, \quad \sigma_t = \sigma_0 t + (1 - t)$$

avec $\boldsymbol{\mu}_0$ et σ_0 des hyperparamètres. On obtient

$$\mathbf{v}_t(\mathbf{x} \mid \mathbf{x}_1) = \mathbf{x}_1 - \mathbf{x} + \frac{\sigma_0 - 1}{\sigma_0 t + (1 - t)}(\mathbf{x} - t\mathbf{x}_1)$$

Algorithme

Algorithm:

- Initialiser les paramètres $\boldsymbol{\theta}$ du modèle $\mathbf{v}_{t,\boldsymbol{\theta}}(\mathbf{x})$
- Pour i de 1 à N_{iter}:
  - Échantillonner $\mathbf{x}_1$ de la distribution cible p_{target}
  - Échantillonner $\mathbf{x}_0$ de la distribution simple p_0
  - Échantillonner t uniformément dans $[0, 1]$
  - Calculer $\mathbf{x}_t = \mu_t(\mathbf{x}_1) + \sigma_t(\mathbf{x}_1)\mathbf{x}_0$
  - Calculer le champ conditionnel $\mathbf{v}_t(\mathbf{x}_t \mid \mathbf{x}_1)$
  - Calculer la perte

$$\mathcal{L} = \| \mathbf{v}_{t,\boldsymbol{\theta}}(\mathbf{x}_t) - \mathbf{v}_t(\mathbf{x}_t \mid \mathbf{x}_1) \|^2$$

  - Mettre à jour les paramètres $\boldsymbol{\theta}$ via une étape de descente de gradient sur $\mathcal{L}$

- On peut proposer plusieurs chemins conditionnels
- On peut traiter le cas conditionnel ou on cherche à générer une distribution conditionnelle $p_{target}(\mathbf{x} \mid \mathbf{c})$

Flot matching pour les PINNs

- Pour les pinns on construisait une densité qui approchait

$h(\mathbf{u_\theta})$ (typiquement le résidu ou le gradient).

- Contrairement au flot normalisé on a pas accès a la densité qu'on construit donc on peut pas minimiser le KL.
- On a bésoin échantillon et comment échantillonner selon $h(\mathbf{u_\theta})$ (qui n'est pas unbe distribution).

Remark: On va utiliser une méthiode de bootstrap a poids qui va nous permettre d'échantillonner selon une densité proportionnelle à $h(\mathbf{u_\theta})$ sans connaitre la densité. Ensuite on fera un entrainement de type flot matching.

- Bootstrap a poids:
  - On échantillonne n points uniformément dans le domaine $\mathbf{x}_i$.
  - On calcule les poids $w_i = \frac{h(\mathbf{u_\theta})(\mathbf{x}_i)}{\sum_{j=1}^{n} h(\mathbf{u_\theta})(\mathbf{x}_j)}$
  - On rééchantillonne n points avec remplacement selon les poids w_i pour obtenir des points $\mathbf{x}_i \sim p_{\theta_p}$
- Comment echantillonner avec les poids
  - on crée des variables cumulatives $c_i = \sum_{j=1}^{i} w_j$
  - on tire $p \in \mathcal{U}[0,1]$ et on cherche i tel que $c_{i-1} \leq p < c_i$

Applications

- Super resolution d'images

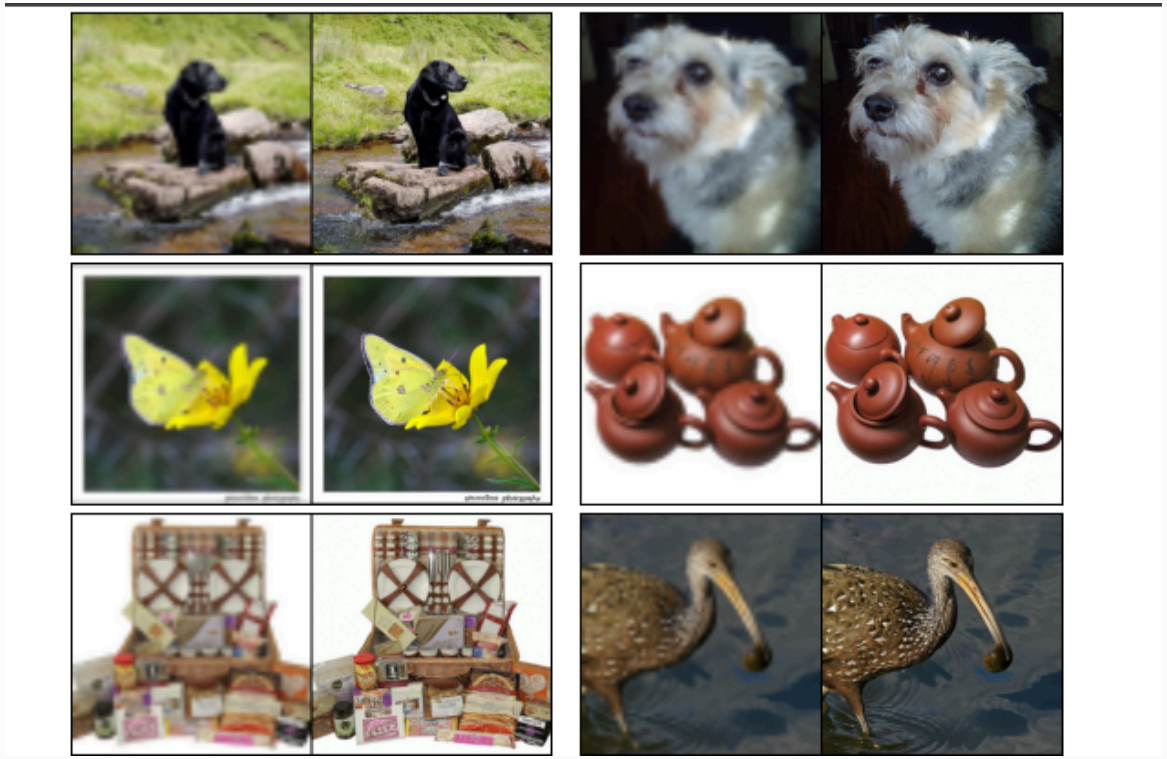


Figure 8: Super resolution d'images avec flow matching. Flow matching paper

- Génération de protéine
- Génération de turbulence etc