Advanced Machine Learning
Normalizing Flow

Alexandre Vérine

Master Intelligence Artificielle, Systèmes, Données

18/02/2022

Program for today

1. Normalizing Flow in theory
   - How does it work ?
   - What can we do with it ?
   - How is it trained ?
2. Structure of Normalizing flow
   - Theoretical Structures
   - First Structures
   - Coupling - Glow
   - Residual Network - ResFlow
3. Current Limits of Normalizing Flow

Overview

A Normalizing Flow is usually seen as:

- a *generative model*,
- a *bijective mapping*,
- an *invertible neural network*,
- a *density estimator*.

Overview

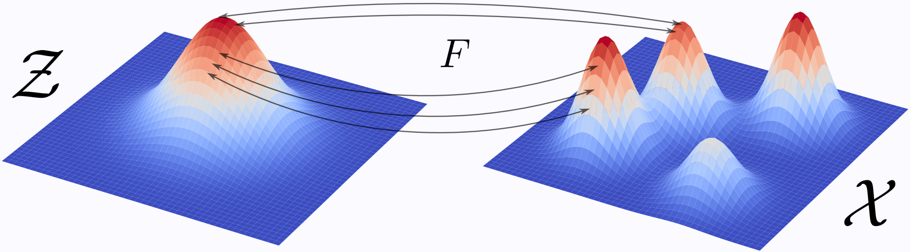

Figure: A mapping between two probability distributions
Point to point

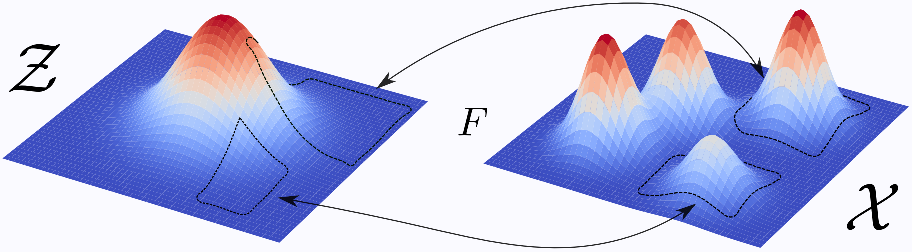

Figure: A mapping between two probability distributions
Subset to subset

Normalizing Flow

A Normalizing Flow is a bijective function between a data space $\mathcal{X}$ and a latent space $\mathcal{Z}$, both subset of $\mathbb{R}^d$.

$$\begin{array}{rccl} F: & \mathcal{X} & \longmapsto & \mathcal{Z} \\ & x & \longmapsto & z = F(x) \end{array}$$

Data and Latent Distributions

In theory, a NF maps a target distribution P, ie the data distribution to a simple latent distribution Q.

Usually, Q is set to be a Normal Gaussian multivariate distribution $\mathcal{N}(0_d, I_d)$. p and q are respectivelly the probability densities of P and Q.

In practice, the mapping is *not perfect*. P^* induces a distribution Q and similarly, the latent distribution Q induces $\widehat{P}$, which is is the learned distribution. The forward pass F is called the *Normalizing* direction while the inverse pass F^{-1} is called the *Generative* direction.

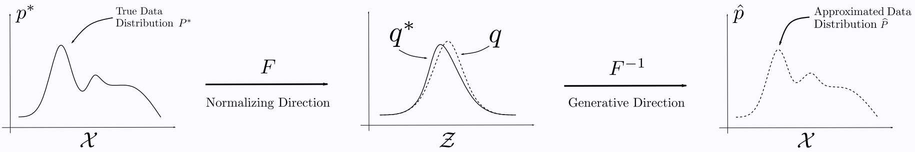

Figure: 1D Normalizing Flow process.

Induced Probabilities ?

Change of Variable Formula

For a bijective and continuous fonction F and a latent distribution Q, the distribution induced by Q and F is defined through the *change of variable formula*:

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \hat{p}(\boldsymbol{x}) = |\det \mathrm{Jac}_F(\boldsymbol{x})|\, q(F(\boldsymbol{x})). \tag{1}$$

Induced Probabilities ?

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \hat{p}(\boldsymbol{x}) = |\det \mathrm{Jac}_F(\boldsymbol{x})|\, q(F(\boldsymbol{x})).$$

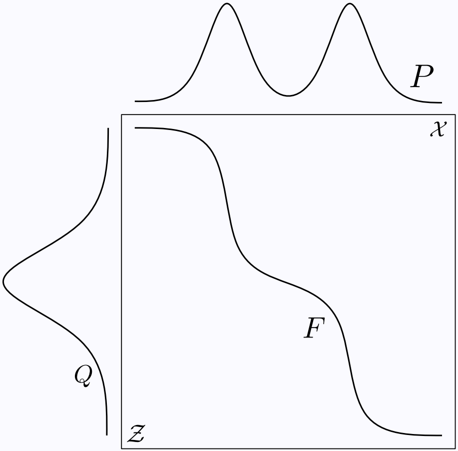

Figure: Example of 1D mapping

Induced Probabilities ?

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \hat{p}(\boldsymbol{x}) = |\det \mathrm{Jac}_F(\boldsymbol{x})|\, q(F(\boldsymbol{x})).$$

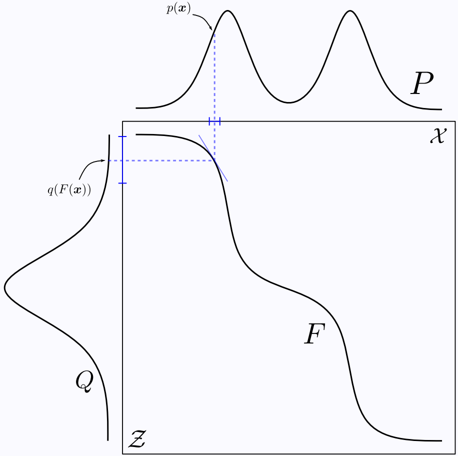

Figure: Example of 1D mapping

Induced Probabilities ?

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \hat{p}(\boldsymbol{x}) = |\det \mathrm{Jac}_F(\boldsymbol{x})| \, q(F(\boldsymbol{x})).$$

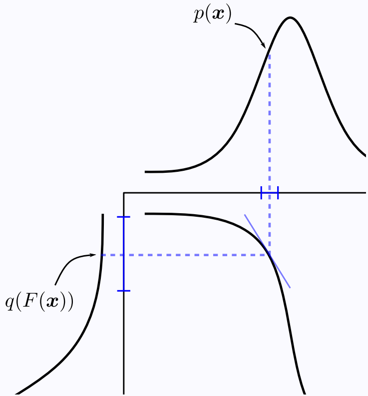

Figure: Example of 1D mapping

Induced Probabilities ?

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \hat{p}(\boldsymbol{x}) = |\det \mathrm{Jac}_F(\boldsymbol{x})|\, q(F(\boldsymbol{x})).$$

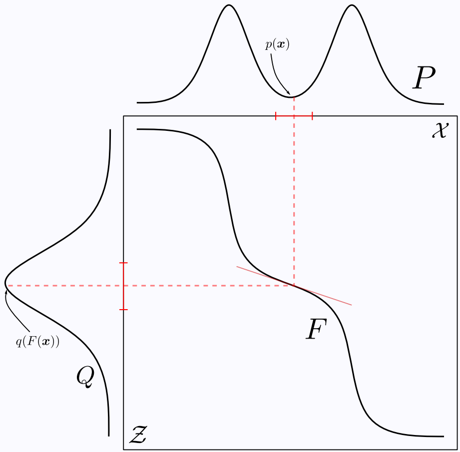

Figure: Example of 1D mapping

$$\forall \boldsymbol{x} \in \mathcal{X}, \quad \hat{p}(\boldsymbol{x}) = |\det \mathrm{Jac}_F(\boldsymbol{x})|\, q(F(\boldsymbol{x})).$$

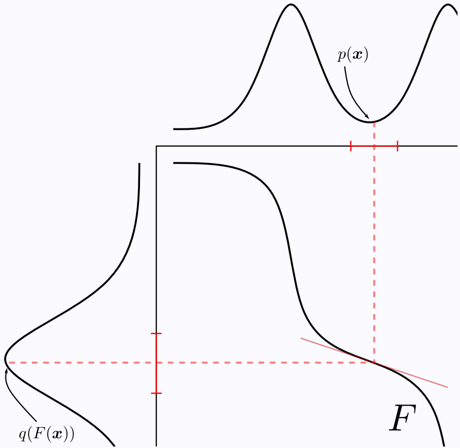

Figure: Example of 1D mapping

To perfom density estimation:

1. Draw $\boldsymbol{x} \sim P^*$,
2. Compute $F(\boldsymbol{x})$ and $|\det \mathrm{Jac}_F(\boldsymbol{x})|$,
3. Compute $\widehat{p}(\boldsymbol{x}) = q(F(\boldsymbol{x}))|\det \mathrm{Jac}_F(\boldsymbol{x})|$.

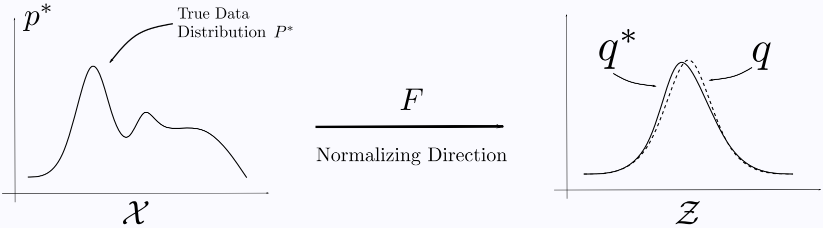

Figure: 1D Normalizing Process of Density Estimation.

To perform data generation:

1. Draw $\boldsymbol{z} \sim Q$,
2. Compute $\boldsymbol{x} = F^{-1}(\boldsymbol{x})$.

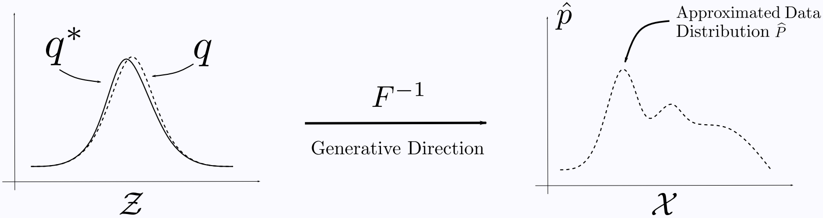

Figure: 1D Normalizing Flow process of Generation.

How is it trained ? - Generative models

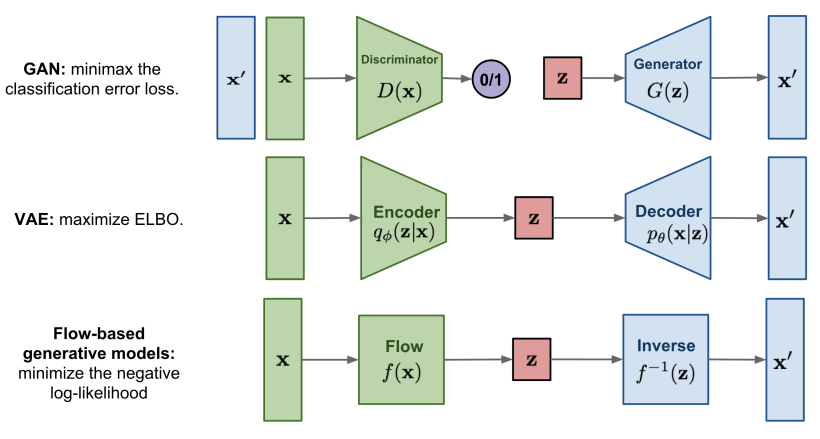

Figure: Different generative models [5]

Log-Likelihood

Loss

The objective is to approximate P^* with $\widehat{P}$. We can minimize the Kullback-Leiber Divergence :

$$\theta = \arg\min_{\theta} \mathcal{D}_{\mathrm{KL}}(P^* \| \widehat{P}).$$

This is equivalent to maximizing the log likelihood :

$$\theta = \arg\max_{\theta} \mathbb{E}_{x \sim \mathcal{X}} \left[\log \widehat{p}(x) \right] .$$

Log-Likelihood

$$\mathcal{D}_{\mathrm{KL}}(P^* \| \widehat{P}) = \int_{\mathcal{X}} p^*(\boldsymbol{x}) \log \left(\frac{p^*(\boldsymbol{x})}{\widehat{p}(\boldsymbol{x})} \right) d\boldsymbol{x}$$

$$\mathrm{nll} = -\mathbb{E}_{\boldsymbol{x} \sim \mathcal{X}} \left[\log \widehat{p}(\boldsymbol{x}) \right]$$

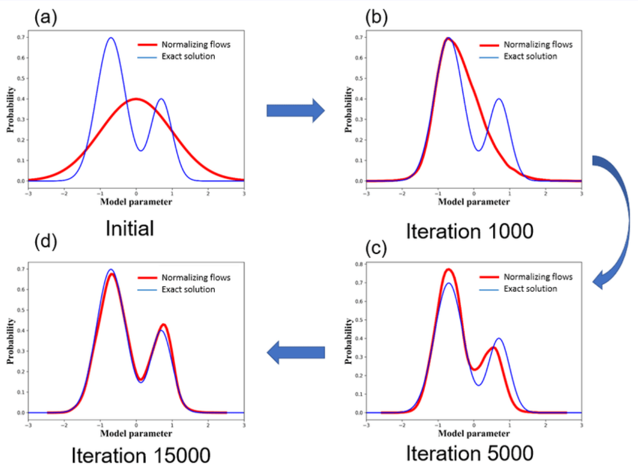

Figure: Learning Process for a 1D Normalizing Flow [6].

Bits/dim

The entropy can be used as an intuitive understanding of the loss. We have :

$$H(P^*, \widehat{P}) = \mathcal{D}_{\mathrm{KL}}(P^* \| \widehat{P}) + H(P^*) \geq H(P^*).$$

In practice, the entropy per dimension in used to compare different models on a same dataset :

$$\mathrm{bpd} = \frac{\sum_{\boldsymbol{x} \in \mathcal{X}} \log\left(\widehat{p}(\boldsymbol{x})/2^K\right)}{d} = \frac{\mathrm{nll}}{dK \log(2)}$$

where K is the number of diffferent bits to encode the data. For instance, we usually have $256 = 2^8$ diffents values per pixel per channel, thus $K = 8$.

Informal definition

Normalizing Flow

A Normalizing Flow is a Neural Network for which we can efficiently compute :

- The forward pass F, ie the normalizing direction,
- the inverse pass F^{-1}, ie the generative direction,
- the determinant of the Jacobian matrix Jac_F for every $\boldsymbol{x}$.

A normalizing Flow is a Neural Network

Usually, a Neural Net is represented as the combination of atomic function $f_i : \boldsymbol{x} \mapsto \sigma(\boldsymbol{W}_i \boldsymbol{x} + \boldsymbol{b}_i)$ such that $F = f_n \circ \cdots \circ f_0$.

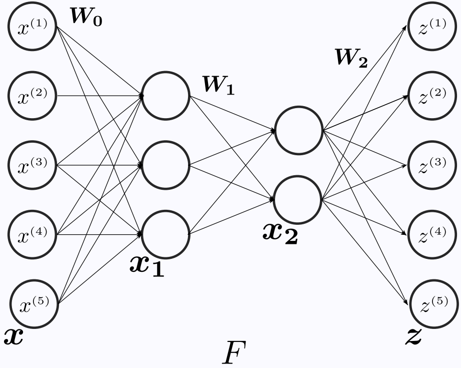

Figure: Representation of a Neural Network.

A normalizing Flow is a Neural Network

For a Normalizing Flow, every atomic block f_i satisfies the properties of F, i.e. bijective and the Jacobian must be efficiently computable.

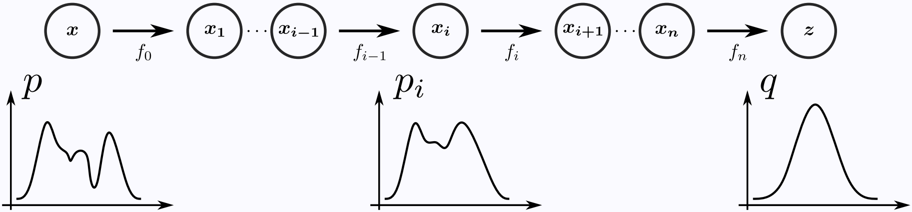

Figure: Atomic representation of the Normalizing Flow.

Therefore, we have:

$$\log \det \mathrm{Jac}_F(\boldsymbol{x}) = \sum_{i=0}^{n} \log \det \mathrm{Jac}_{f_i}(\boldsymbol{x}_i)$$

.

A normalizing Flow is a Neural Network

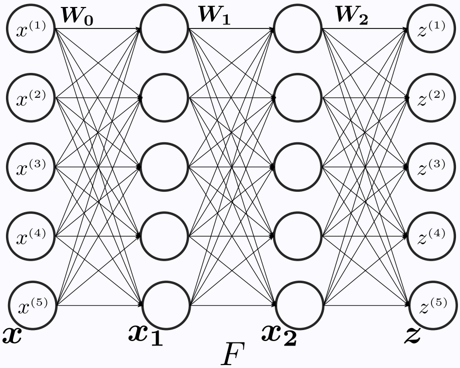

Figure: Representation of an Invertible Neural Network.

Elementwise Flow

Elementwise Flow

The element wise flows are based on elementwise bijective non linear scalar function. Let $h_i : \mathbb{R} \mapsto \mathbb{R}$ be the scalar value bijection :

$$f_i(x_i^1, \dots, x_i^d) = \left(h\left(x_i^1\right), \dots, h\left(x_i^d\right)\right)^T$$

Determinant of the Jacobian matrix

Here, $\mathrm{Jac}_{f_i}(\boldsymbol{x}) = \mathrm{Diag}\left(\frac{dh}{dx}|_{x=x_1}, \dots, \frac{dh}{dx}|_{x=x_d}\right)$, so $|\det \mathrm{Jac}_{f_i}(\boldsymbol{x})| = \prod_i^d \frac{dh}{dx}|_{x=x_i}$.

Limits

- No mixing of variables [2].

Linear Flows

Linear Flows

Let $\boldsymbol{A}_i$ be an invertible matrix of $\mathbb{R}^{d\times d}$ and $\boldsymbol{b}_i$ a vector of $\mathbb{R}^d$. A Linear Flow is based on the linear operation :

$$f_i(\boldsymbol{x}) = \boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i$$

Determinant of the Jacobian matrix

Here, $\mathrm{Jac}_{f_i}(\boldsymbol{x}) = \boldsymbol{A}_i$, so $|\det \mathrm{Jac}_{f_i}(\boldsymbol{x})| = |\det \boldsymbol{A}_i| \neq 0$.

Limits

Poor Expressiveness.

Planar Flows

Planar Flows

Let $h : \mathbb{R} \mapsto \mathbb{R}$, h' its derivative, $\boldsymbol{u}$ and $\boldsymbol{w}$ two vectors of $\mathbb{R}^d$ and a scalar b. The Planar flow is based on atomic blocks f_i such that:

$$f_i(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}h\left(\boldsymbol{w}^T\boldsymbol{x} + b\right)$$

Determinant of the Jacobian matrix

Here, $\mathrm{Jac}_{f_i}(\boldsymbol{x}) = \boldsymbol{I}_d + \boldsymbol{u}h'(\boldsymbol{w}^T\boldsymbol{x} + b)\boldsymbol{w}^T$, so
$|\det \mathrm{Jac}_{f_i}(\boldsymbol{x})| = 1 + h'(\boldsymbol{w}^T\boldsymbol{x} + b)\boldsymbol{u}^T\boldsymbol{w})$.

Limits

No closed form of the inverse.

Radial Flows

Radial Flows

Let $\boldsymbol{x}_0$ a vector of $\mathbb{R}^d$, α and β two scalars. The Radial Flow is based on atomic blocks f_i such that:

$$f_i(\boldsymbol{x}) = \boldsymbol{x} + \frac{\beta}{\alpha + \|\boldsymbol{x} - \boldsymbol{x}_0\|}(\boldsymbol{x} - \boldsymbol{x}_0)$$

Determinant of the Jacobian matrix

Here, $|\det \mathrm{Jac}_{f_i}(\boldsymbol{x})| = \left[1 + \frac{\beta}{\alpha+\|\boldsymbol{x}-\boldsymbol{x}_0\|}\right]^{d-1}\left[1 + \frac{\beta}{\alpha+\|\boldsymbol{x}-\boldsymbol{x}_0\|} + \frac{\|\boldsymbol{x}-\boldsymbol{x}_0\|}{(\alpha+\|\boldsymbol{x}-\boldsymbol{x}_0\|)^2}\right]$.

Limits

No closed form of the inverse.

Radial and Planar Flows

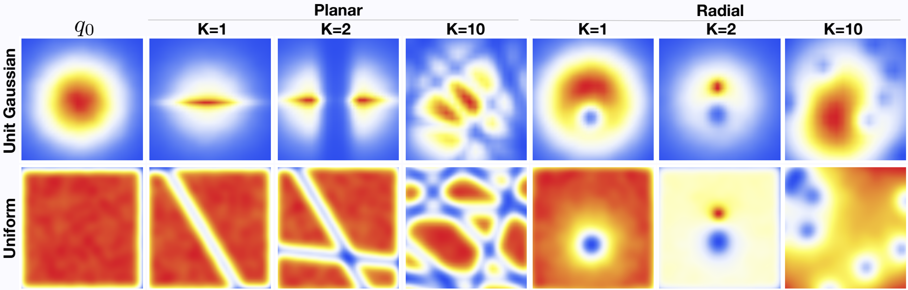

Figure: Example of Planar and Radial Flows [3].

Coupling Function

A coupling function is coupling disjoint partitions of the dataset. For an input $\boldsymbol{x} \in \mathbb{R}^d$, we consider a split $(\boldsymbol{x}^A, \boldsymbol{x}^B) \in \mathbb{R}^k \times \mathbb{R}^{d-k}$. Usually, $k = d/2$. We consider a bijective function, generally elementwise, $\boldsymbol{h} : \boldsymbol{x} \mapsto \boldsymbol{h}(x, \theta) \in \mathbb{R}^k$ parameterized by θ. We can train a small neural network such that θ becomes a function of $\boldsymbol{x}^A$ or $\boldsymbol{x}^B$. We can therefore define the atomic coupling function :

$$f_i(\begin{bmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^B \end{bmatrix}) = \begin{bmatrix} \boldsymbol{y}^A \\ \boldsymbol{y}^B \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}^A, g(\boldsymbol{x}^B)) \\ \boldsymbol{x}^B \end{bmatrix}.$$

The reverse is :

$$\boldsymbol{f}_i^{-1}(\begin{bmatrix} \boldsymbol{y}^A \\ \boldsymbol{y}^B \end{bmatrix}) = \begin{bmatrix} \boldsymbol{x}^A \\ \boldsymbol{x}^B \end{bmatrix} = \begin{bmatrix} \boldsymbol{h}^{-1}(\boldsymbol{y}^B, g(\boldsymbol{x}^B)) \\ \boldsymbol{x}^B \end{bmatrix}.$$

Coupling Function

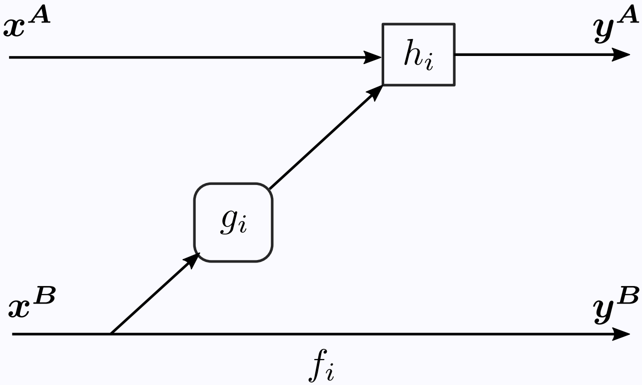

Figure: Atomic coupling block.

Coupling Function

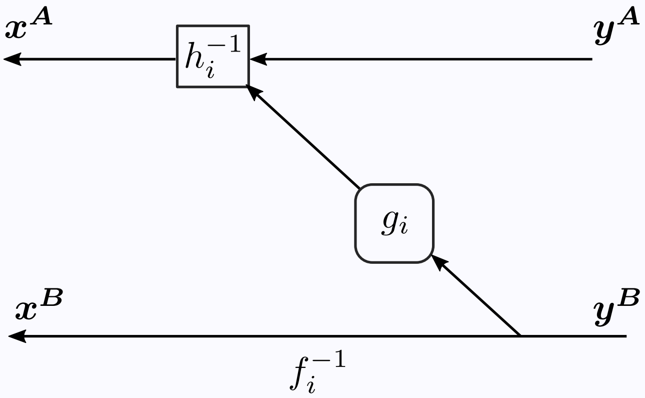

Figure: Inverse Atomic coupling block.

Coupling Function

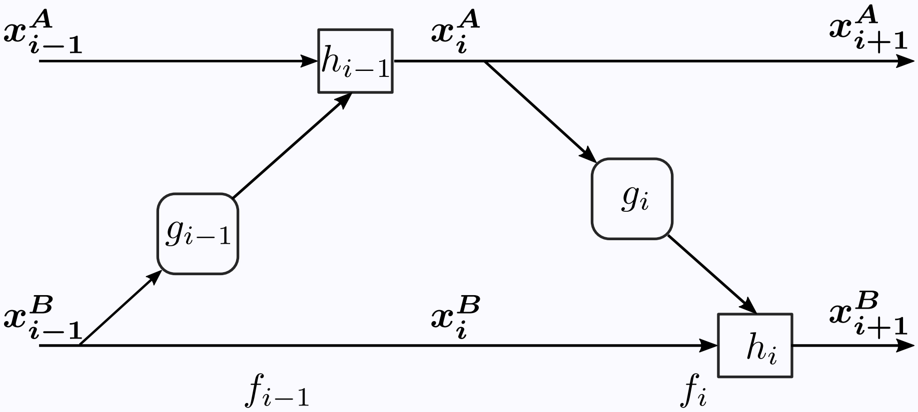

Figure: Composed Atomic coupling block.

Coupling Function

Derterminant of the Jacobian Matrix

Here, $\mathrm{Jac}_{f_i}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial h}{\partial \boldsymbol{x}^A}\big|_{\boldsymbol{x}^A, g(\boldsymbol{x}^B)} & \frac{\partial h}{\partial \boldsymbol{x}^B}\big|_{\boldsymbol{x}^A, g(\boldsymbol{x}^B)} \\ \boldsymbol{0} & \boldsymbol{I}_k \end{pmatrix}$, thus

$|\det \mathrm{Jac}_{f_i}(\boldsymbol{x})| = |\frac{\partial h}{\partial \boldsymbol{x}^A}(\boldsymbol{x}^A)|$

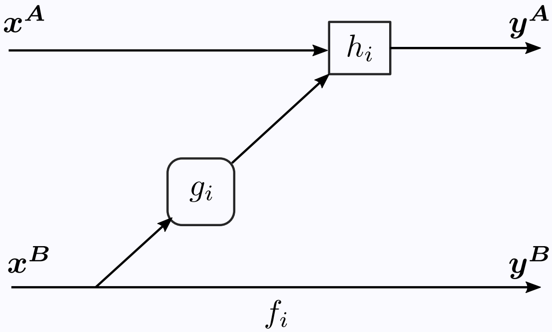

Figure: Atomic coupling block.

Additive Coupling

Additive Coupling Flow

The additive coupling flow has atomic function defined with an elementwise fucntion $h_i : \mathbb{R} \mapsto \mathbb{R}$ and the scalar function $g_i : \mathbb{R}^d \mapsto \mathbb{R}$:

$$h_i(x_i) = x_i + g_i(\boldsymbol{x}^B).$$

Therefore,

$$f_i\left(\begin{bmatrix}\boldsymbol{x}^A \\ \boldsymbol{x}^B\end{bmatrix}\right) = \begin{bmatrix}\boldsymbol{x}_A + g(\boldsymbol{x}^B) \\ \boldsymbol{x}^B\end{bmatrix}.$$

Determinant of the Jacobian matrix

Here, $|\det \mathrm{Jac}_{f_i}| = 1$. The flow is said to be volume preserving.

Limits

Very deep structure and poor expressiveness.

Affine Coupling

Affine Coupling Flow

The affine coupling flow has atomic function defined with an elementwise fucntion $h_i : \mathbb{R} \mapsto \mathbb{R}$ and the scalar function $g_i : \mathbb{R}^d \mapsto \mathbb{R}$:

$$h_i(x_i) = g_i^1(\boldsymbol{x}^B)x_i + g_i^2(\boldsymbol{x}^B).$$

Therefore,

$$f_i\left(\begin{bmatrix}\boldsymbol{x}^A \\ \boldsymbol{x}^B\end{bmatrix}\right) = \begin{bmatrix}g^1(\boldsymbol{x}^B)\boldsymbol{x}_A + g^2(\boldsymbol{x}^B) \\ \boldsymbol{x}^B\end{bmatrix}.$$

Determinant of the Jacobian matrix

Here, $|\det \mathrm{Jac}_{f_i}| = \left(g_i^1(\boldsymbol{x}^B)\right)^k$.

Limits

Very deep structure.

Residual Flow

Residual Flow

A residual flow is composed of residual atomic blocks f_i defined with function $\boldsymbol{g}_i : \mathbb{R}^d \mapsto \mathbb{R}^d$ such that:

$$f_i(\boldsymbol{x}) = x_i + \boldsymbol{g}(x_i).$$

In $\boldsymbol{g}$, we can stack Convolutional layers, batchnorms, activations, etc...

Determinant of the Jacobian Matrix

Here, the determinant of the log determinant of the Jacobian matrix is estimated by a Russian Roulette Estimator, and a Skilling-Hutchingson trace estimator [1]:

$$\log \det \mathrm{Jac}_{f_i}(vx) = \mathbb{E}_{n\sim \mathrm{Geo}(p), v\sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})} \left[\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} \frac{\boldsymbol{v}^T \left[\mathrm{Jac}_g(\boldsymbol{x})^k \boldsymbol{v}\right]}{p(n \geq k)} \right]$$

Inverse

A residual Network is invertible only if $Lip(g_i) < 1$. We can compute the inverse with the Inverse-point algorithm:

Algorithm 1 Inverse via fixed-point iteration.

$\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}_i$
while NotConverged **do**
  $\boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}_i - \boldsymbol{g}(\boldsymbol{x}_{i-1})$
end while

Limits

Interative inverse and approximated Jacobian.

Lipschitz Limitation

Since Normalizing Flows are diffeomorphism, there are bi-Lipschitz :

$$\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \quad \|F(\boldsymbol{x}_1) - F(\boldsymbol{x}_2)\|_2 \leq L_1 \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2,$$

and

$$\forall \boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{Z}, \quad \|F^{-1}(\boldsymbol{z}_1) - F^{-1}(\boldsymbol{z}_2)\|_2 \leq L_2 \|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2.$$

Therefore, some pathological target distributions can be highlighted [4].

Let F be L_1-Lipschitz. Then:

$$\mathrm{TV}(P^*, \widehat{P}) \geq \sup_{R, \boldsymbol{x}_0} \left(P^*(B_{R,\boldsymbol{x}_0}) - \frac{\gamma\left(\frac{d}{2}, \frac{L_1^2 R^2}{2}\right)}{\Gamma\left(\frac{d}{2}\right)} \right)$$

Therefore, if we find a ball for which the true measure satisfies $P^*(B_{R,\boldsymbol{x}_0}) > \gamma(\frac{d}{2}, \frac{L_1^2 R^2}{2})/\Gamma(\frac{d}{2})$, then the TV is necessarily strictly positive.

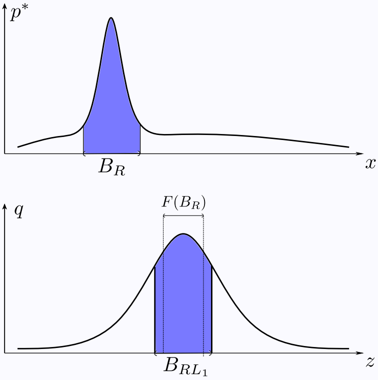

Figure: Example of a pathological target distribution: the subset B_R concentrates most of the weight in $P^*(B_R)$, but $\widehat{P}(B_R) = Q(F(B_R))$ can only be as large as $Q(B_{R_{L_1}})$.

Let F^{-1} be L_2-Lipschitz. We consider the balls centered on $F^{-1}(0)$, we have the lower bound:

$$\mathrm{TV}(P^*, \widehat{P}) \geq \sup_R \left(\frac{\gamma\left(\frac{d}{2}, \frac{R^2}{2L_2^2}\right)}{\Gamma\left(\frac{d}{2}\right)} - P^*(B_{R,F^{-1}(0)}) \right)$$

Therefore, if we find a ball for which the the true measure satisfies $P^*(B_{R,F^{-1}(0)}) < \frac{\gamma(d/2, R^2/2L_2^2)}{\Gamma(d/2)}$, then the TV is necessarily strictly positive.

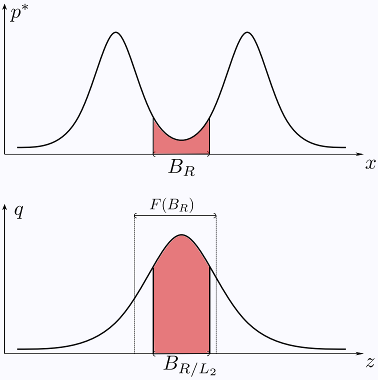

Figure: Example of a pathological target distribution: the subset B_R concentrates little weight in $P^*(B_R)$, but $\widehat{P}(B_R) = Q(F(B_R))$ can only be as small as $Q(B_R/L_2)$.

Conclusion

Normalizing Flow are a very powerfull tools to perform :

- Data generation,
- Density estimation.

Reversible networks have a larger spectrum of applications such as learning physics models, manifold learning...
However, they suffer from a expensive computational cost for training and inference. We will see that in the Practical session that can be found on : `https://github.com/AlexVerine/AdvancedML_NF`.

References I

[1] Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual Flows for Invertible Generative Modeling. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.*, July 2020. arXiv: 1906.02735.

[2] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. arXiv: 1908.09257.

[3] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *arXiv:1505.05770 [cs, stat]*, June 2016.

[4] Alexandre Verine, Yann Chevaleyre, Fabrice Rossi, and Benjamin Negrevergne. On the expressivity of bi-Lipschitz normalizing flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*, June 2021.

References II

[5] Lilian Weng. Flow-based deep generative models. *lilianweng.github.io/lil-log*, 2018.

[6] Xuebin Zhao, Andrew Curtis, and Xin Zhang. Bayesian Variational Seismic Tomography using Normalizing Flows. In *EGU General Assembly Conference Abstracts*, EGU General Assembly Conference Abstracts, pages EGU21–1455, April 2021.