Unconstrained Monotonic Neural Networks

Antoine Wehenkel & Gilles Louppe
antoine.wehenkel@uliege.be

What? UMNN is a new neural network architecture for modelling monotonic functions.

How? The strictly positive scalar output of a neural network is numerically integrated.

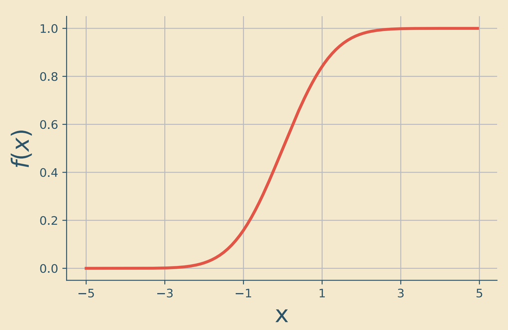
Applications? We combine UMNNs with autoregressive flows to perform density estimation.

Code

Arxiv 1908.05164

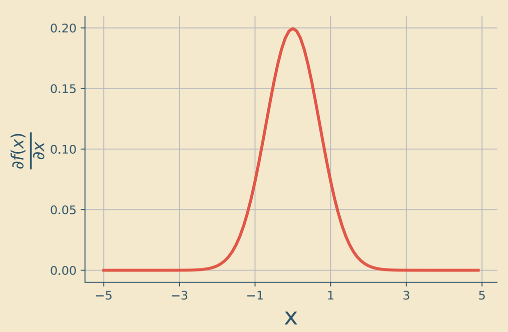
Monotonic Networks (UMNN)

A necessary and sufficient condition for a continuously derivable function to be strictly monotonic is for its derivative to be of constant sign.



Derivation →

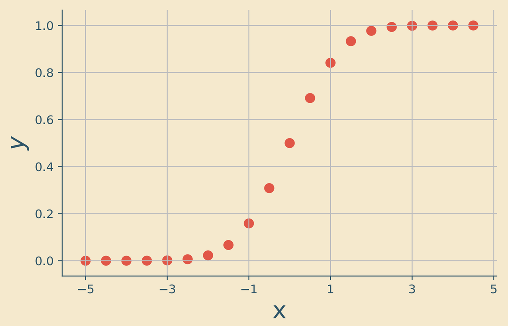
← Integration



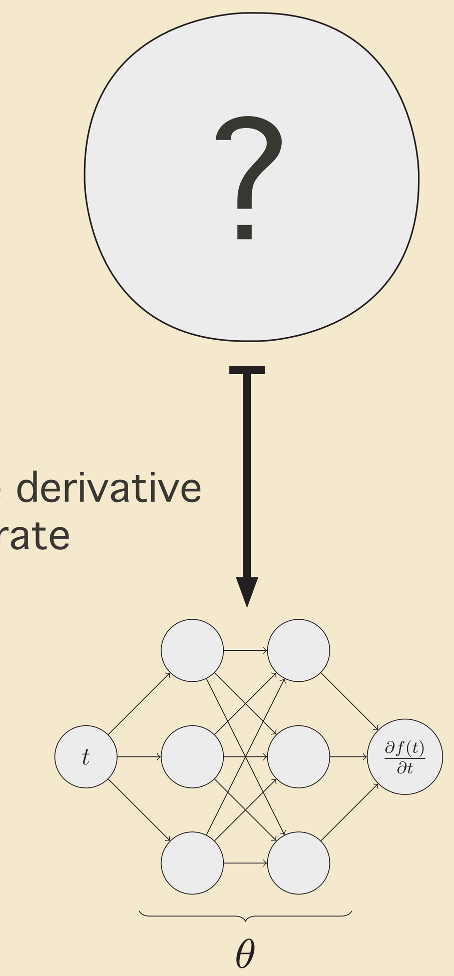
Monotonicity is hard to enforce

BUT

positiveness is not!



Derivation →



Solution:
Model the derivative and integrate

$$\hat{y} = F(x;\theta) = \int_0^x \frac{\partial f(t;\theta)}{\partial t} dt + \text{const}$$

← Forward: Integration

Update

$$\mathcal{L}_\theta = \sum_i l(\hat{y}_i, y_i)$$

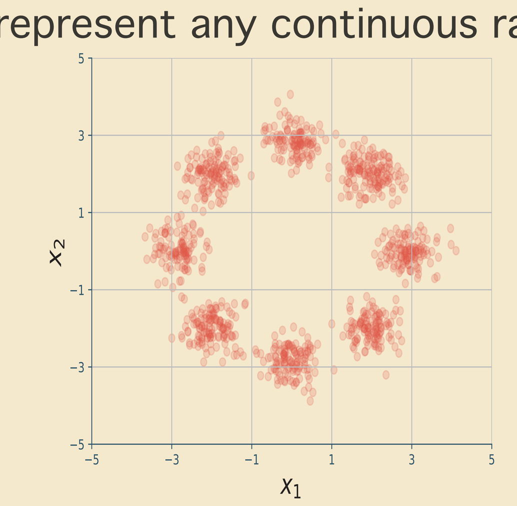
Backward: Integration →

$$\frac{d\mathcal{L}_\theta}{d\theta} = \sum_i \frac{\partial l(\hat{y}_i, y_i)}{\partial \hat{y}_i}\frac{d\hat{y}_i}{d\theta}$$

$$\frac{d\hat{y}_i}{d\theta} = \frac{dF(x;\theta))}{d\theta} = \int_0^x \frac{d\left(\frac{\partial f(t;\theta)}{\partial t}\right)}{d\theta} dt$$

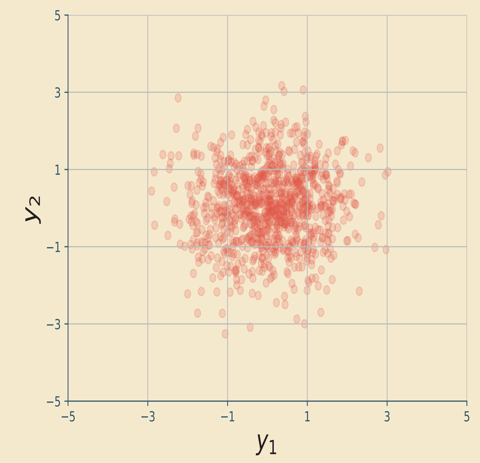
Change of variables

The adequate bijective function combined with any base distribution is able to represent any continuous random variable.



Density Estimation →

← Sampling



$p(\mathbf{x};\theta) = p_\mathbf{y}(\mathbf{y} = g(\mathbf{x};\theta))\,|\det J_g(\mathbf{x};\theta)|$, $\quad g(\cdot, \theta)$ an invertible neural network.

Autoregressive Density Estimation

Autoregressive transformations can be written as

$$g(\mathbf{x};\theta) = [g_1(x_1;\theta) \ldots g_i(\mathbf{x}_{1:i};\theta) \ldots g_d(\mathbf{x}_{1:d};\theta)].$$

Invertible? Yes, if each scalar transformation in $\mathbf{h}^i$ is itself invertible.

$$x_1 = g_1^{-1}(y_1;\theta)$$
$$x_i = g_i^{-1}(y_i, \mathbf{x}_{1:i-1};\theta)$$

The induced multivariate density can be expressed by the chain rule as

$$p(\mathbf{x};\theta) = p(x_1;\theta)\prod_{i=1}^{d-1} p(x_{i+1}|\mathbf{x}_{1:i};\theta).$$

We build UMNN into an autoregressive transformation as

$$g^i(\mathbf{x}_{1:i};\theta) = F^i(x_i, \mathbf{h}^i(\mathbf{x}_{1:i-1})) = \int_0^{x_i} f^i(t, \mathbf{h}^i(\mathbf{x}_{1:i-1}))\,dt + \beta^i(\mathbf{h}^i(\mathbf{x}_{1:i-1}))$$

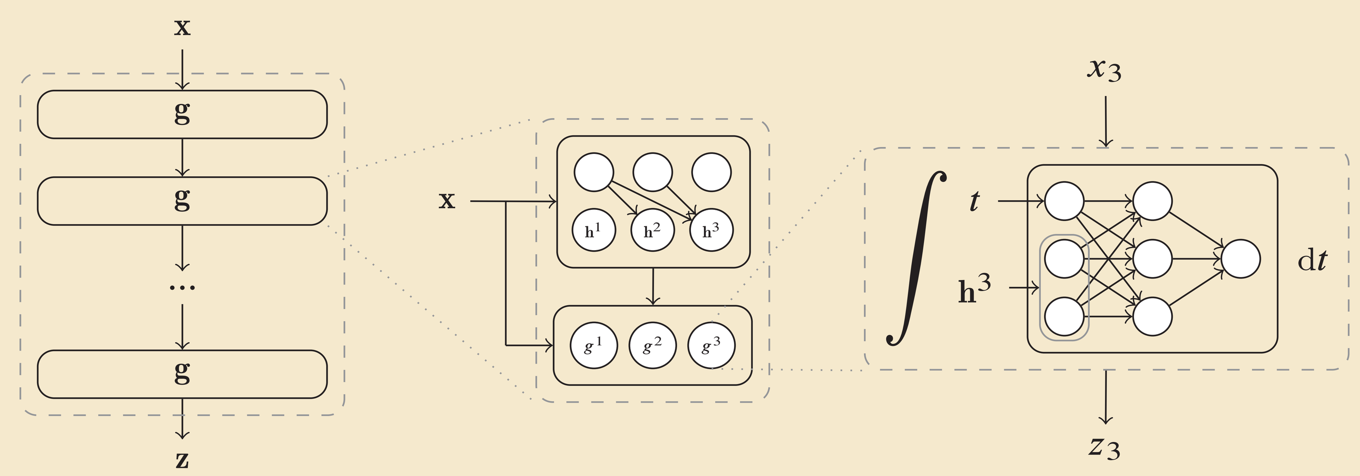
The autoregressive embeddings $\mathbf{h}^i$ are computed with a masked autoregressive network.

We call this transformation UMNN-MAF, it leads to the following elegant expression of the log-likelihood of a point given the model:

$$\log p(\mathbf{x};\theta) = \log p_\mathbf{y}(\mathbf{g}(\mathbf{x};\theta)) + \sum_{i=1}^{d} \log f^i(x_i, \mathbf{h}^i(\mathbf{x}_{1:i-1})).$$

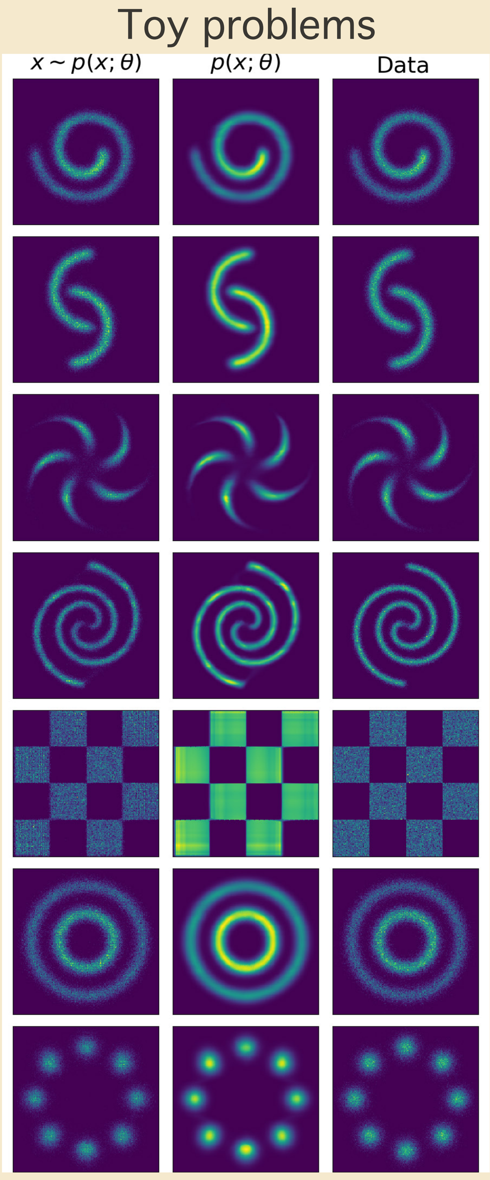
Architecture

The combination of a UMNN with autoregressive transformations can be visualized as follows:



Results

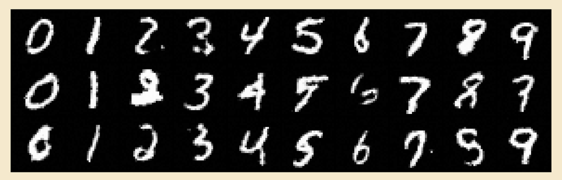
Toy problems



Density estimation

	Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST
(a)	RealNVP - Dinh et al. [2017]	$-0.17_{\pm.01}$	$-8.33_{\pm.14}$	$18.71_{\pm.02}$	$13.55_{\pm.49}$	$-153.28_{\pm1.78}$	-
	Glow - Kingma and Dhariwal [2018]	$-0.17_{\pm.01}$	$-8.15_{\pm.40}$	$19.92_{\pm.08}$	$11.35_{\pm.07}$	$-155.07_{\pm.03}$	-
	FFJORD - Grathwohl et al. [2018]	$\underline{-0.46}_{\pm.01}$	$\underline{-8.59}_{\pm.12}$	$\underline{14.92}_{\pm.08}$	$\underline{10.43}_{\pm.04}$	$\underline{-157.40}_{\pm.19}$	-
(b)	MADE - Germain et al. [2015]	$3.08_{\pm.03}$	$-3.56_{\pm.04}$	$20.98_{\pm.02}$	$15.59_{\pm.50}$	$-148.85_{\pm.28}$	$2.04_{\pm.01}$
	MAF - Papamakarios et al. [2017]	$-0.24_{\pm.01}$	$-10.08_{\pm.02}$	$17.70_{\pm.02}$	$11.75_{\pm.44}$	$-155.69_{\pm.28}$	$1.89_{\pm.01}$
	TAN - Oliva et al. [2018]	$\underline{-0.60}_{\pm.01}$	$\underline{\mathbf{-12.06}}_{\pm.02}$	$\mathbf{13.78}_{\pm.02}$	$\underline{11.01}_{\pm.48}$	$\underline{\mathbf{-159.80}}_{\pm.07}$	$\underline{1.19}$
(c)	NAF - Huang et al. [2018]	$-0.62_{\pm.01}$	$-11.96_{\pm.33}$	$15.09_{\pm.40}$	$\underline{\mathbf{8.86}}_{\pm.15}$	$-157.73_{\pm.30}$	-
	B-NAF - De Cao et al. [2019]	$-0.61_{\pm.01}$	$\underline{-12.06}_{\pm.09}$	$14.71_{\pm.38}$	$8.95_{\pm.07}$	$-157.36_{\pm.03}$	-
	SOS - Jaini et al. [2019]	$-0.60_{\pm.01}$	$-11.99_{\pm.41}$	$15.15_{\pm.1}$	$8.90_{\pm.11}$	$-157.48_{\pm.41}$	1.81
	UMNN-MAF (ours)	$\underline{\mathbf{-0.63}}_{\pm.01}$	$-10.89_{\pm.7}$	$\underline{13.99}_{\pm.21}$	$9.67_{\pm.13}$	$\underline{-157.98}_{\pm.01}$	$\underline{\mathbf{1.13}}_{\pm.02}$

MNIST



Take home messages

- Any monotonic function can be modeled by a neural network that represents the function derivative.
- The backward pass is memory efficient thanks to Leibniz rule.
- UMNNs can be used as building blocks of autoregressive bijective maps and provide a state of the art density estimator.