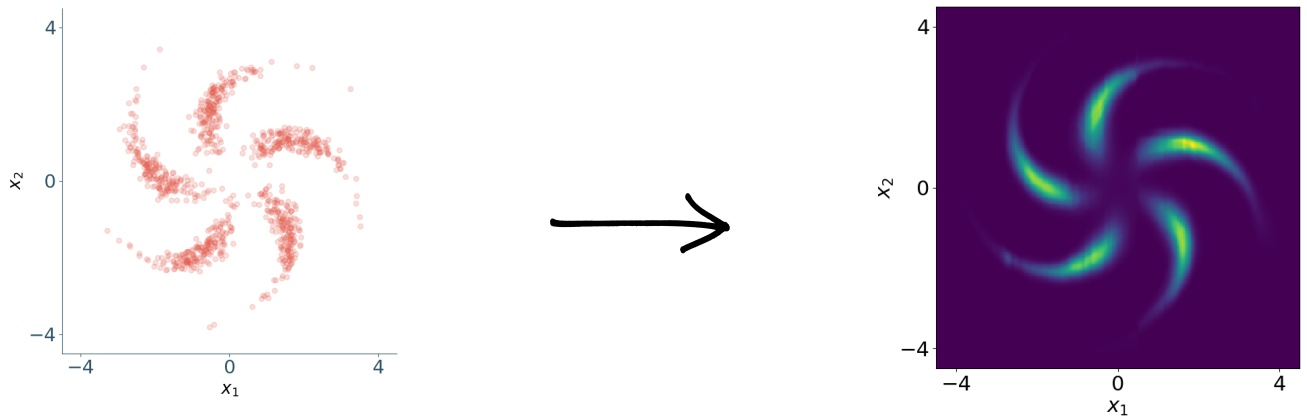
Unconstrained Monotonic Neural Networks

Accepted @ Neural Information Processing Systems 2019

Benelearn 2019

Antoine Wehenkel Gilles Louppe

Density Estimation 👇



> Density estimation aims at estimating the pdf of underlying data from an iid dataset.

Applications:

- Anomaly Detection.
- Out Of Distribution detection.
- Sampling.

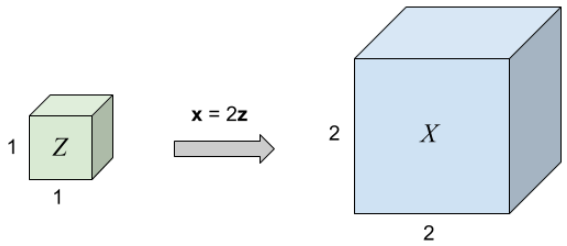
Change of Variables Theorem (1)

> Given a random variable $\mathcal{Z}$ and a bijective function f, how does the density of $\mathcal{X} = f(\mathcal{Z})$ behave in terms of $p(\mathbf{z})$ and f?

Change of Variables Theorem (1)

> Given a random variable $\mathcal{Z}$ and a bijective function f, how does the density of $\mathcal{X} = f(\mathcal{Z})$ behave in terms of $p(\mathbf{z})$ and f?

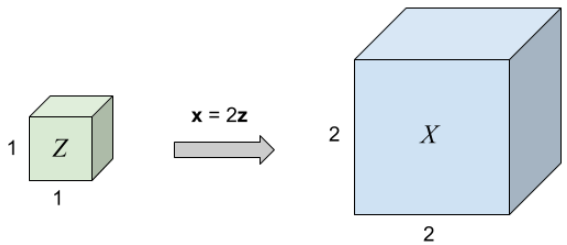
Assume $p(z)$ is a uniformely distributed unit cube in $\mathbb{R}^3$, and $\mathbf{x} = f(\mathbf{z}) = 2\mathbf{z}$.



Change of Variables Theorem (1)

> Given a random variable $\mathcal{Z}$ and a bijective function f, how does the density of $\mathcal{X} = f(\mathcal{Z})$ behave in terms of $p(\mathbf{z})$ and f?

Assume $p(z)$ is a uniformely distributed unit cube in $\mathbb{R}^3$, and $\mathbf{x} = f(\mathbf{z}) = 2\mathbf{z}$.



The total probability mass must be conserved, therefore

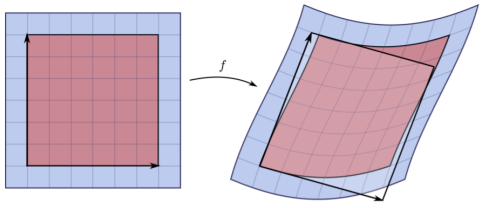
$p(\mathbf{x} = f(\mathbf{z})) = p(\mathbf{z})\frac{V_\mathbf{z}}{V_\mathbf{x}} = p(\mathbf{z})\frac{1}{8}$, where $\frac{1}{8} = \left|\det \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}\right|^{-1}$ is the determinant of the linear transformation f.

———

Change of Variables Theorem (2)

What if the transformation is non linear?

- The Jacobian $J_f(\mathbf{z})$ of $\mathbf{x} = f(\mathbf{z})$ represents the infinitesimal linear transformation in the neighbourhood of $\mathbf{z}$.



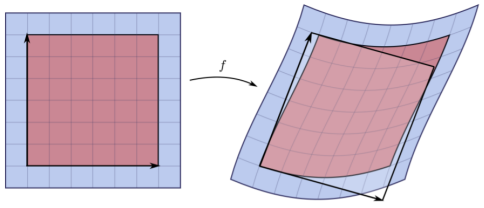
———

https://www.lesswrong.com/posts/KKwv9kcQz29vqPLAD/a-primer-on-matrix-calculus-part-2-jacobians-and-other-fun

Change of Variables Theorem (2)

What if the transformation is non linear?

- The Jacobian $J_f(\mathbf{z})$ of $\mathbf{x} = f(\mathbf{z})$ represents the infinitesimal linear transformation in the neighbourhood of $\mathbf{z}$.



- If the function is bijective map then the mass must be conserved locally.

Therefore, we can compute the local change of density as

$$p(\mathbf{x}) = p(\mathbf{z}) \left|\det J_f(\mathbf{z})\right|^{-1} .$$

———

https://www.lesswrong.com/posts/KKwv9kcQz29vqPLAD/a-primer-on-matrix-calculus-part-2-jacobians-and-other-fun

Change of Variables Theorem (3)

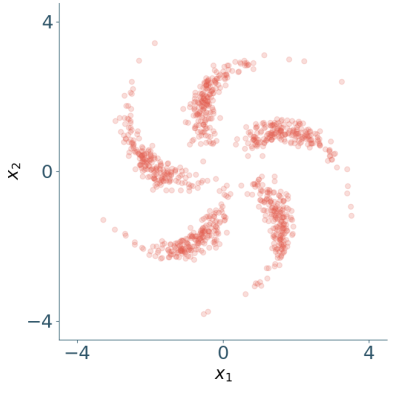
The combination of the right bijective map and any base distribution allows to represent any continuous random variable.

Change of Variables Theorem (3)

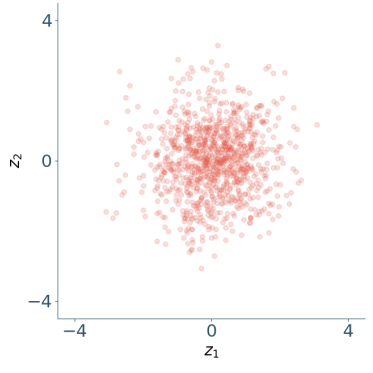
The combination of the right bijective map and any base distribution allows to represent any continuous random variable.

$$p(\mathbf{x};\theta) = p(\mathbf{z} = \mathbf{g}(\mathbf{x};\theta)) \left|\det J_g(\mathbf{x};\theta)\right|, \quad \mathbf{g}(.;\theta) \text{ a neural network.}$$

- The bijective function takes in samples and maps them to noise.
- This process is refered as normalization if the noise distribution is normal.

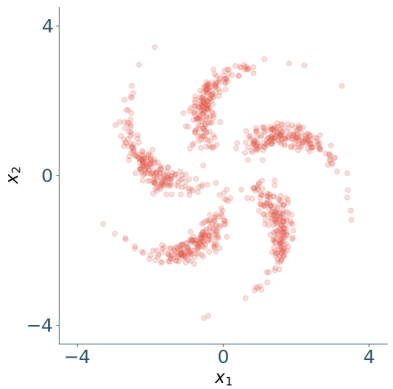


Density Estimation

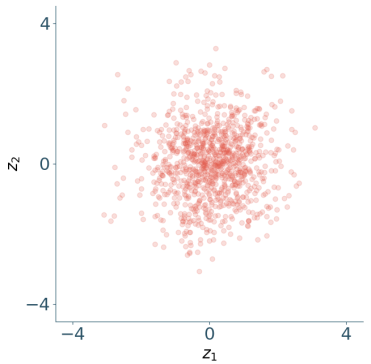


Change of Variables Theorem (3)

The combination of the right bijective map and any base distribution allows to represent any continuous random variable.



Sampling



Once learned, the function can be inverted in order to generate samples.

Bijectivity with Neural Nets? 🤔

- $\begin{bmatrix} z_1 & \dots & z_d \end{bmatrix} = g(\begin{bmatrix} x_1 & \dots & x_d \end{bmatrix})$, g can be a NN.
- g is autoregressive if it can be decomposed as: $z_i = g_i(\begin{bmatrix} x_1 & \dots & x_i \end{bmatrix})$
- If the g_i are invertible with respect to $x_i \forall i$, g is bijective.

Bijectivity with Neural Nets? 🤔

- $\begin{bmatrix} z_1 & \dots & z_d \end{bmatrix} = g(\begin{bmatrix} x_1 & \dots & x_d \end{bmatrix})$, g can be a NN.
- g is autoregressive if it can be decomposed as: $z_i = g_i(\begin{bmatrix} x_1 & \dots & x_i \end{bmatrix})$
- If the g_i are invertible with respect to $x_i \forall i$, g is bijective.

The determinant of the Jacobian can be efficiently computed.

The Jacobian of an autoregressive tranformation has the following form:

$$J_g(\mathbf{x}) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & 0 & 0 \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & 0 \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} \end{bmatrix}.$$

Bijectivity with Neural Nets? 🤔

- $\begin{bmatrix} z_1 & \dots & z_d \end{bmatrix} = g(\begin{bmatrix} x_1 & \dots & x_d \end{bmatrix})$, g can be a NN.
- g is autoregressive if it can be decomposed as: $z_i = g_i(\begin{bmatrix} x_1 & \dots & x_i \end{bmatrix})$
- If the g_i are invertible with respect to $x_i \forall i$, g is bijective.

The determinant of the Jacobian can be efficiently computed.

The Jacobian of an autoregressive tranformation has the following form:

$$J_g(\mathbf{x}) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \frac{\partial g_2}{\partial x_3} \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & 0 & 0 \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & 0 \\ \frac{\partial g_3}{\partial x_1} & \frac{\partial g_3}{\partial x_2} & \frac{\partial g_3}{\partial x_3} \end{bmatrix}.$$

Chain Rule

An autoregressive density estimator learns the chain rule's factors:
$p(\mathbf{x}) = p(x_1)\Pi_{i=2}^{d} p(x_i|x_1, ..., x_{i-1}).$

Example: Masked Autoregressive Networks

> Idea: Autoregressive Networks combined with linear transformations.

- $z_1 = \sigma_1 \times x_1 + \mu_1$
- $z_i = \sigma_i(x_1, ..., x_{i-1}) \times x_i + \mu_i(x_1, ..., x_{i-1})$

Example: Masked Autoregressive Networks

> Idea: Autoregressive Networks combined with linear transformations.

- $z_1 = \sigma_1 \times x_1 + \mu_1$
- $z_i = \sigma_i(x_1, ..., x_{i-1}) \times x_i + \mu_i(x_1, ..., x_{i-1})$

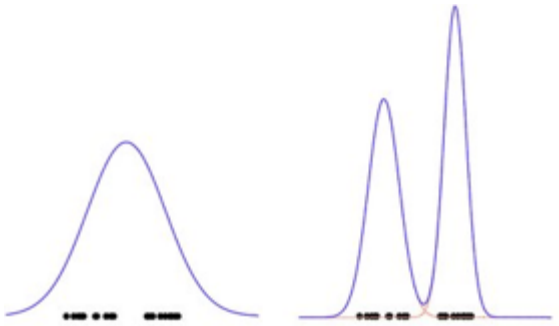
Invertible?

- $x_1 = g_1^{-1}(\begin{bmatrix} z_1 & ... & z_d \end{bmatrix}) = g_1^{-1}(z_1) = \frac{(z_1 - \mu_1)}{\sigma_1}$
- $x_i = \frac{z_i - \mu_i(\begin{bmatrix} x_1 & ... & x_{i-1} \end{bmatrix})}{\sigma_i(\begin{bmatrix} x_1 & ... & x_{i-1} \end{bmatrix})}$

Example: Masked Autoregressive Networks

> Idea: Autoregressive Networks combined with linear transformations.

But linear transformations are not very expressive:

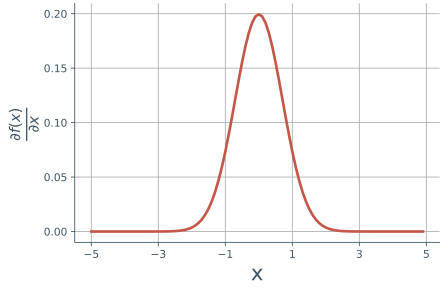


Monotonic Transformations

> How can we enforce the monotonicity of a function modeled by a neural network

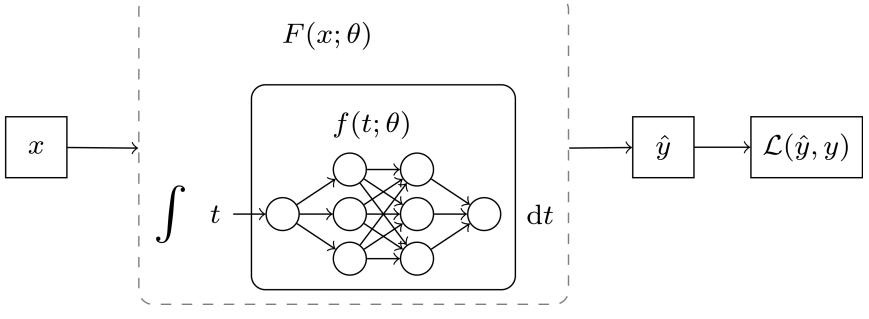
UMNN: Unconstrained Monotonic Neural Network

Our solution: To model and integrate the derivative.



The only constraint is on the output value which must be of constant sign (e.g. positive).

This can be achieved by applying an exponential on the output neuron.



Learning of UMNN 🤯

How can we backward through the numerical integrator?

By keeping track of all the computations and
building the corresponding computation graph.

Learning of UMNN 🤯

How can we backward through the numerical integrator?

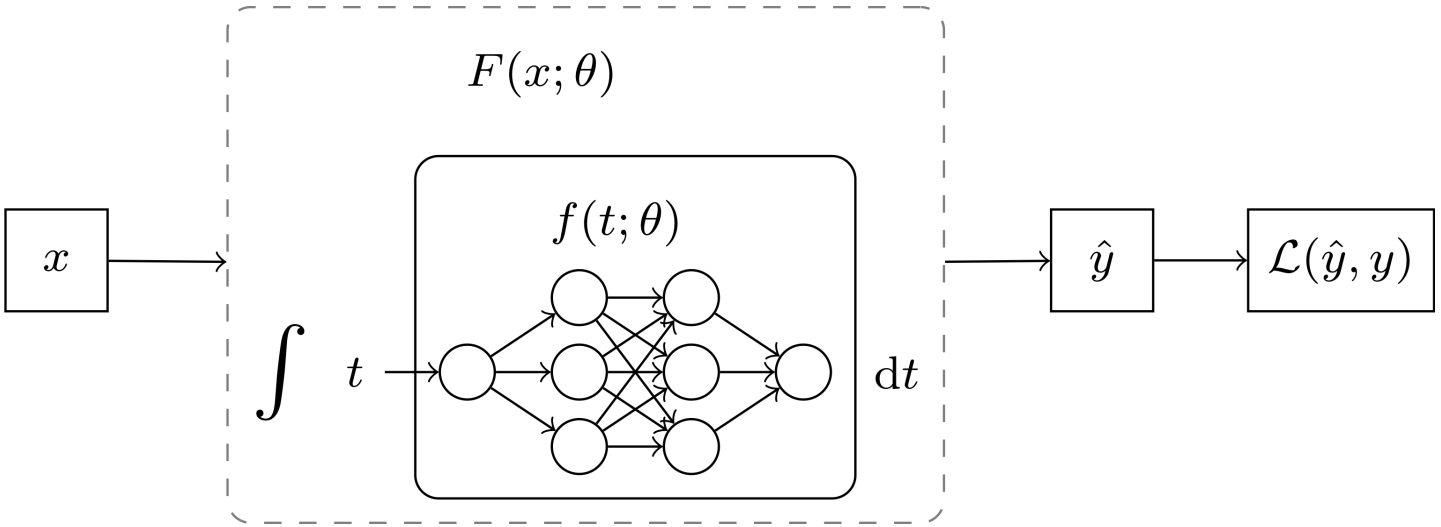
By keeping track of all the computations and building the corresponding computation graph.

Learning of UMNN 🤯

How can we backward through the numerical integrator?

By thanking Leibniz for his rule and applying it.

$$\frac{\mathrm{d}}{\mathrm{d}\omega}\left(\int_{a(\omega)}^{b(\omega)} f(t;\omega)dt\right) = f(b(\omega);\omega)\frac{\mathrm{d}}{\mathrm{d}\omega}b(\omega) - f(a(\omega);\omega)\frac{\mathrm{d}}{\mathrm{d}\omega}a(\omega) + \int_{a(\omega)}^{b(\omega)} \frac{\partial}{\partial\omega} f(t;\omega)dt$$

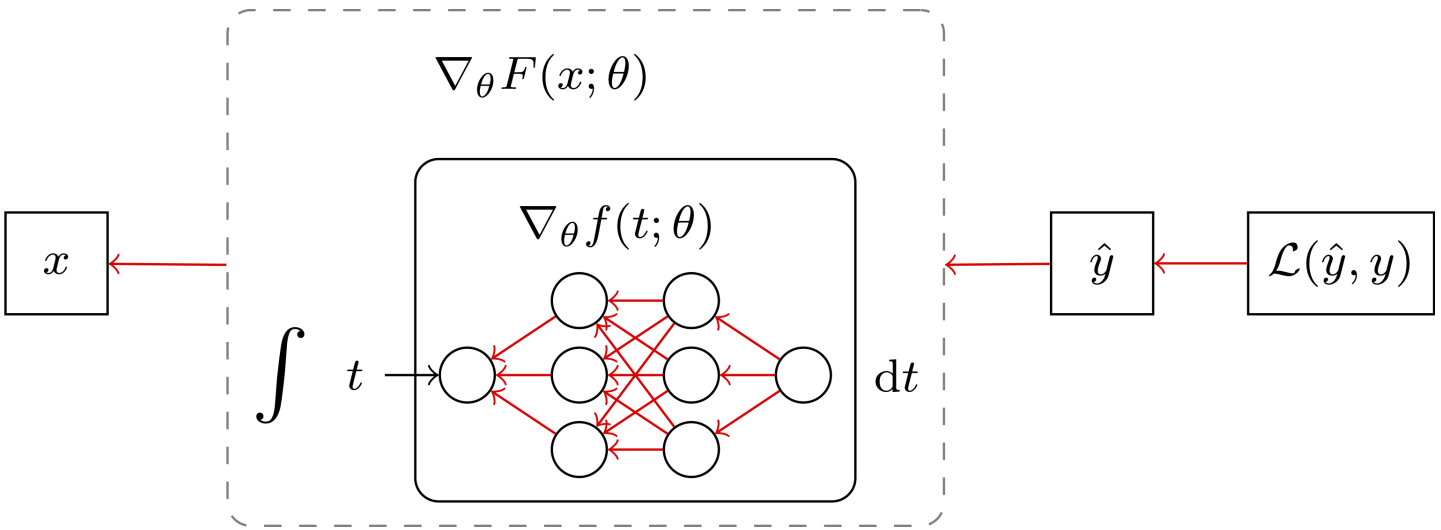


Learning of UMNN 🤯

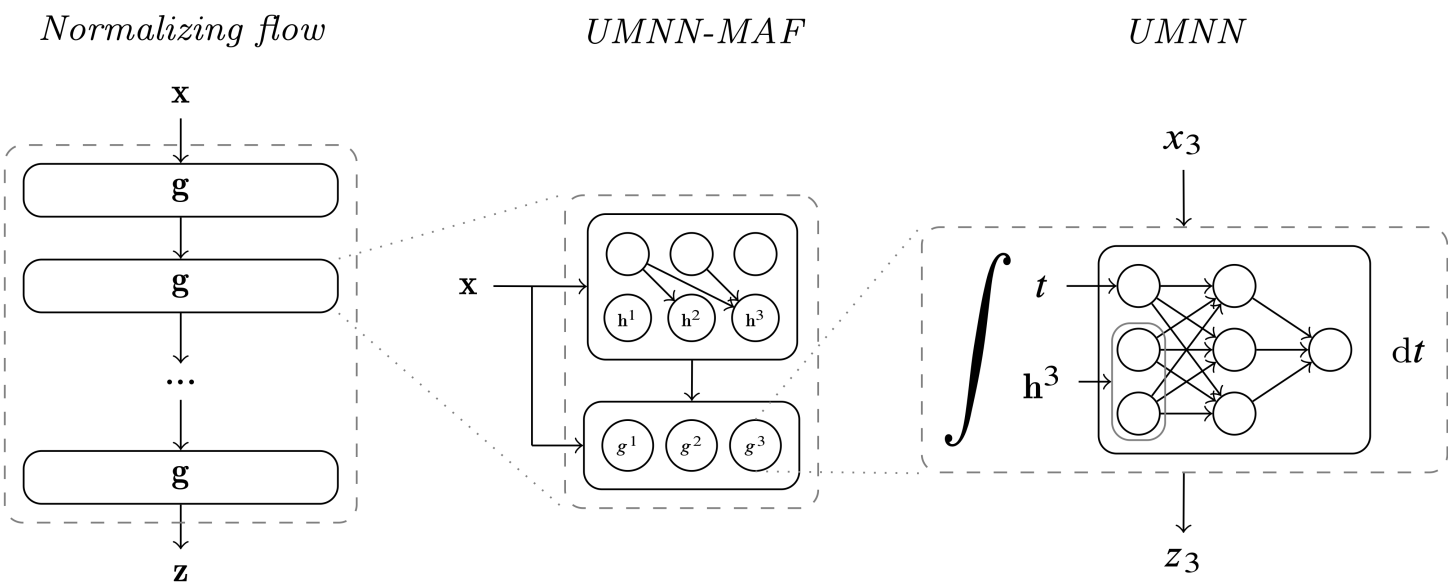
How can we backward through the numerical integrator?

By thanking Leibniz for his rule and applying it.

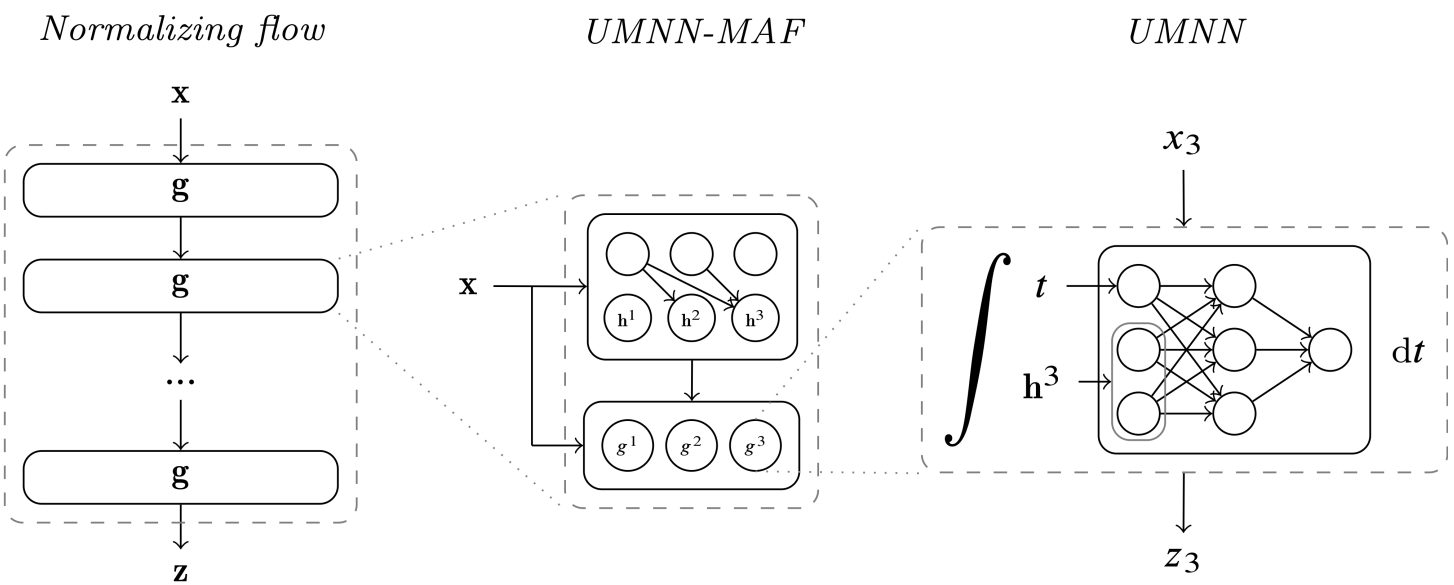
$$\frac{\mathrm{d}}{\mathrm{d}\omega}\left(\int_{a(\omega)}^{b(\omega)} f(t;\omega)dt\right) = f(b(\omega);\omega)\frac{\mathrm{d}}{\mathrm{d}\omega}b(\omega) - f(a(\omega);\omega)\frac{\mathrm{d}}{\mathrm{d}\omega}a(\omega) + \int_{a(\omega)}^{b(\omega)} \frac{\partial}{\partial\omega} f(t;\omega)dt$$



Density Estimation with UMNN: UMNN-MAF



Density Estimation with UMNN: UMNN-MAF

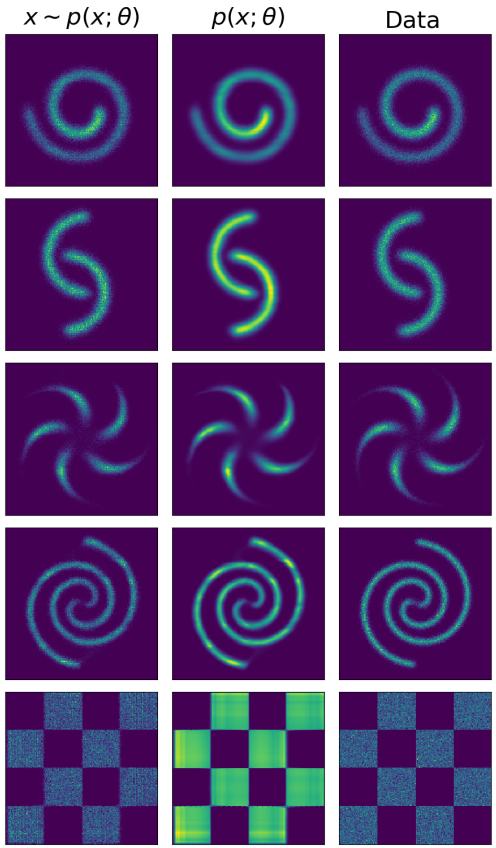


The Jacobian has the following form:

$$J_F(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \frac{\partial F_1}{\partial x_3} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \frac{\partial F_2}{\partial x_3} \\ \frac{\partial F_3}{\partial x_1} & \frac{\partial F_3}{\partial x_2} & \frac{\partial F_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} f_1(x_1) & 0 & 0 \\ \frac{\partial F_2}{\partial x_1} & f_2(x_2; x_1) & 0 \\ \frac{\partial F_3}{\partial x_1} & \frac{\partial F_3}{\partial x_2} & f_3(x_3; x_1, x_2) \end{bmatrix}$$

Experimental Results

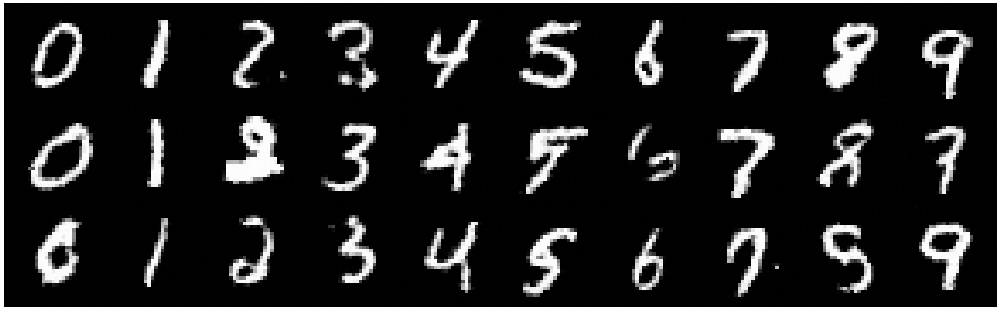
Toy Problems



- Learnt by maximum likelihood on the train samples.
- Data dimensionality is $\mathbf{2}$.
- Multimodality.
- Discontinuity.

Experimental Results

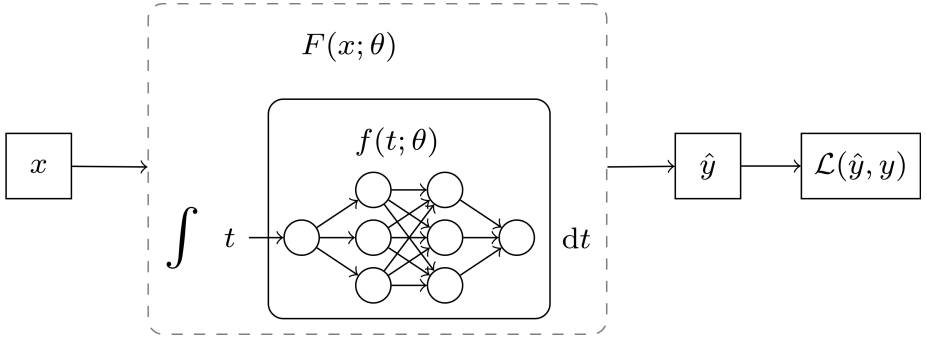
MNIST

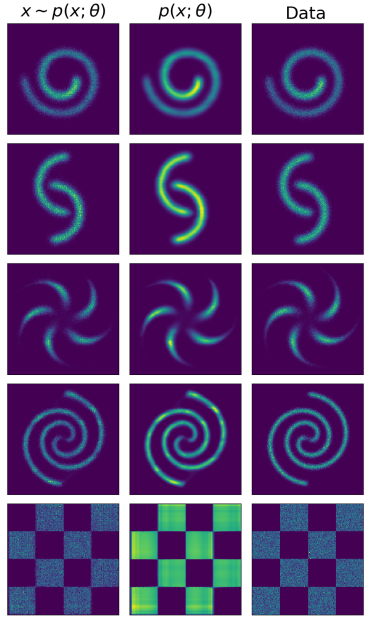


- Learnt by maximum likelihood on the training samples.
- Data dimensionality is 784.
- Samples are generated by sampling noise and inverting the model.
- The model has no knowledge about the structure of the images, i.e.: $p(\mathbf{x}) = p(x_1, ..., x_{784}) = p(x_1)\Pi_{i=2}^{784} p(x_i|x_1, ..., x_{i-1})$

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 the Leibniz rule.
- UMNN can be used as a building block of autoregressive bijective maps and provide a state of the art density estimator.





Fin

Experimental Results

Density Estimation Benchmarks

	Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST
	RealNVP - Dinh et al. [2017]	$-0.17_{\pm.01}$	$-8.33_{\pm.14}$	$18.71_{\pm.02}$	$13.55_{\pm.49}$	$-153.28_{\pm1.78}$	-
(a)	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]	$-0.46_{\pm.01}$	$-8.59_{\pm.12}$	$14.92_{\pm.08}$	$10.43_{\pm.04}$	$-157.40_{\pm.19}$	-
	MADE - Germain et al. [2015]	$3.08_{\pm.03}$	$-3.56_{\pm.04}$	$20.98_{\pm.02}$	$15.59_{\pm.50}$	$-148.85_{\pm.28}$	$2.04_{\pm.01}$
(b)	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]	$-0.60_{\pm.01}$	$\mathbf{-12.06}_{\pm.02}$	$\mathbf{13.78}_{\pm.02}$	$11.01_{\pm.48}$	$\mathbf{-159.80}_{\pm.07}$	1.19
	NAF - Huang et al. [2018]	$-0.62_{\pm.01}$	$-11.96_{\pm.33}$	$15.09_{\pm.40}$	$\mathbf{8.86}_{\pm.15}$	$-157.73_{\pm.30}$	-
(c)	B-NAF - De Cao et al. [2019]	$-0.61_{\pm.01}$	$-12.06_{\pm.09}$	$14.71_{\pm.38}$	$8.95_{\pm.07}$	$-157.36_{\pm.03}$	-
	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)	$\mathbf{-0.63}_{\pm.01}$	$-10.89_{\pm.7}$	$13.99_{\pm.21}$	$9.67_{\pm.13}$	$-157.98_{\pm.01}$	$\mathbf{1.13}_{\pm.02}$

- Learnt by maximum likelihood on the training samples.

Experimental Results

Density Estimation Benchmarks

	Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST
	RealNVP - Dinh et al. [2017]	$-0.17_{\pm.01}$	$-8.33_{\pm.14}$	$18.71_{\pm.02}$	$13.55_{\pm.49}$	$-153.28_{\pm1.78}$	-
(a)	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]	$-0.46_{\pm.01}$	$-8.59_{\pm.12}$	$14.92_{\pm.08}$	$10.43_{\pm.04}$	$-157.40_{\pm.19}$	-
	MADE - Germain et al. [2015]	$3.08_{\pm.03}$	$-3.56_{\pm.04}$	$20.98_{\pm.02}$	$15.59_{\pm.50}$	$-148.85_{\pm.28}$	$2.04_{\pm.01}$
(b)	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]	$-0.60_{\pm.01}$	$\mathbf{-12.06}_{\pm.02}$	$\mathbf{13.78}_{\pm.02}$	$11.01_{\pm.48}$	$\mathbf{-159.80}_{\pm.07}$	1.19
	NAF - Huang et al. [2018]	$-0.62_{\pm.01}$	$-11.96_{\pm.33}$	$15.09_{\pm.40}$	$\mathbf{8.86}_{\pm.15}$	$-157.73_{\pm.30}$	-
(c)	B-NAF - De Cao et al. [2019]	$-0.61_{\pm.01}$	$-12.06_{\pm.09}$	$14.71_{\pm.38}$	$8.95_{\pm.07}$	$-157.36_{\pm.03}$	-
	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)	$\mathbf{-0.63}_{\pm.01}$	$-10.89_{\pm.7}$	$13.99_{\pm.21}$	$9.67_{\pm.13}$	$-157.98_{\pm.01}$	$\mathbf{1.13}_{\pm.02}$

- Learnt by maximum likelihood on the training samples.

Experimental Results

Density Estimation Benchmarks

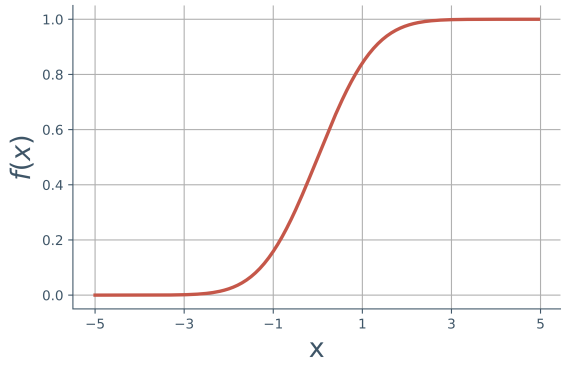
	Dataset	POWER	GAS	HEPMASS	MINIBOONE	BSDS300	MNIST
	RealNVP - Dinh et al. [2017]	$-0.17_{\pm.01}$	$-8.33_{\pm.14}$	$18.71_{\pm.02}$	$13.55_{\pm.49}$	$-153.28_{\pm1.78}$	-
(a)	Glow - Kingma and Dhariwal [2018]	$-0.17_{\pm.01}$	$-8.15_{\pm.40}$	$19.92_{\pm.08}$	$11.35_{\pm.07}$	$-155.07_{\pm.03}$	-
	FFJORD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

- Learnt by maximum likelihood on the training samples.

Monotonic Transformations

How can we build a 1D invertible function? Monotonic transformations.

> How can we enforce the monotonicity of a function modeled by a neural network ?



Possible solution: Strictly positive weights and monotonic activation functions.

But it restrains the architectural choice.