Neural Autoregressive Flows

Chin-Wei Huang, David Krueger, Alexandre Lacoste, Aaron Courville

Presented by

Avideep Mukherjee
Advised by
Prof. Vinay P. Namboodiri

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

January 13, 2020

Outline

- Types of Generative Models
- Normalizing Flows
- Autoregressive Models
- Inverse Autoregressive Flow
- Neural Autoregressive Flow

Types of Generative Models

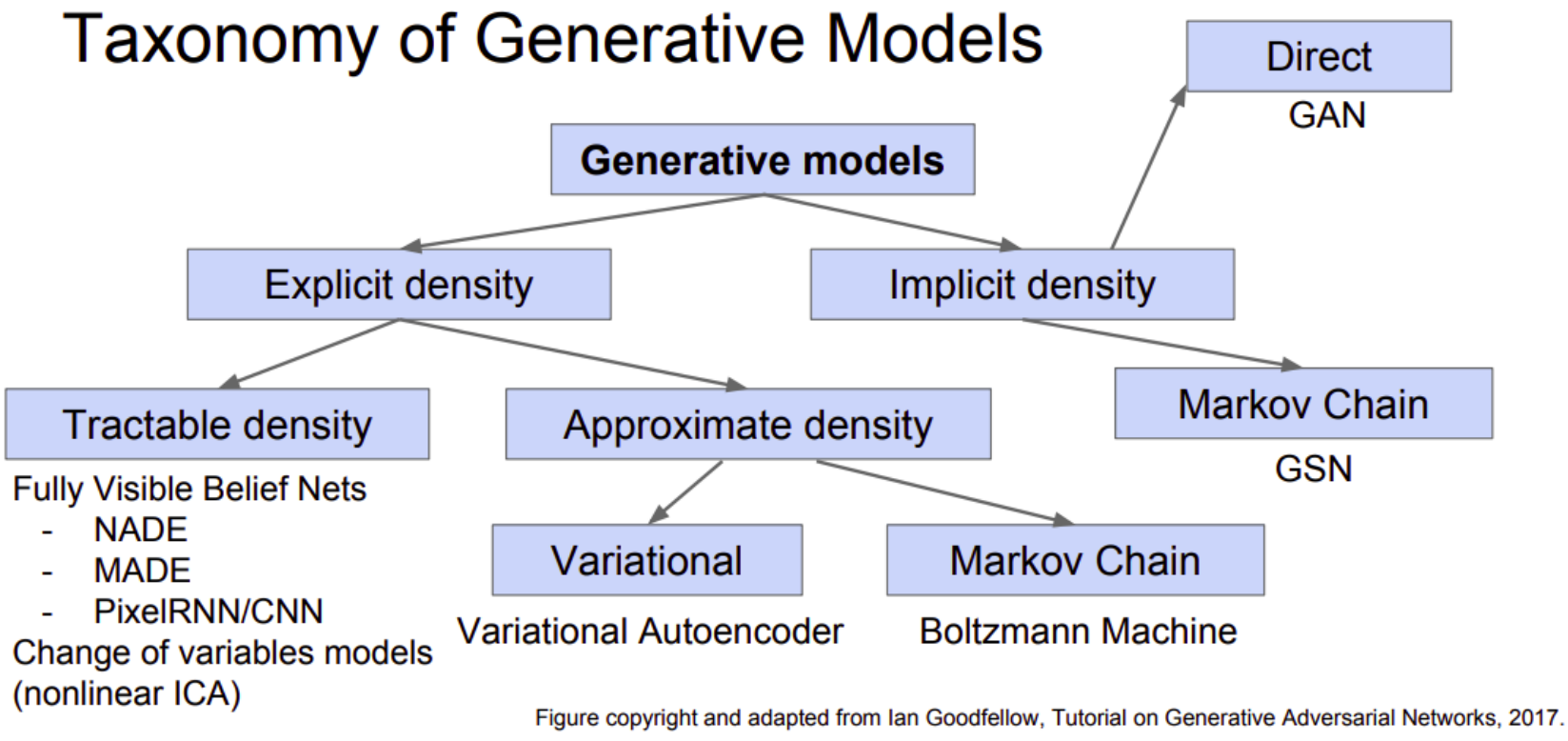


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Normalizing Flows

- Transform an *'easy' paramaterizable base distribution* in a *more complex approximation* for the posterior distribution (Rezende and Mohamed (2015)).

Normalizing Flows

- Transform an *'easy' paramaterizable base distribution* in a *more complex approximation* for the posterior distribution (Rezende and Mohamed (2015)).
- Pass the base distribution through a series (flow) of transformations (*invertible, bijective mappings*).

Normalizing Flows

- Transform an *'easy' paramaterizable base distribution* in a *more complex approximation* for the posterior distribution (Rezende and Mohamed (2015)).
- Pass the base distribution through a series (flow) of transformations (*invertible, bijective mappings*).
- $\mathbf{z} \in \mathbb{R}^d$ and $\mathbf{y} = f(\mathbf{z})$ where $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ Let $\mathbf{z} \sim q(\mathbf{z})$. According to change of variable formula :

Normalizing Flows

- Transform an *'easy' paramaterizable base distribution* in a *more complex approximation* for the posterior distribution (Rezende and Mohamed (2015)).
- Pass the base distribution through a series (flow) of transformations (*invertible, bijective mappings*).
- $\mathbf{z} \in \mathbb{R}^d$ and $\mathbf{y} = f(\mathbf{z})$ where $f : \mathbb{R}^d \mapsto \mathbb{R}^d$ Let $\mathbf{z} \sim q(\mathbf{z})$. According to change of variable formula :

$$q_y(\mathbf{y}) = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{y}} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1} \tag{1}$$

by applying the inverse function theorem, $J_{f^{-1}}(q) = [J_f(p)]^{-1}$

Normalizing Flows

Apply a series of such mappings f_k, $k \in 1, \ldots, K$ with $K \in \mathbb{N}_+$ and obtain a **normalizing flow**.

Normalizing Flows

Apply a series of such mappings f_k, $k \in 1, \ldots, K$ with $K \in \mathbb{N}_+$ and obtain a **normalizing flow**.

$$\mathbf{x} = \mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0), \quad \mathbf{z}_0 \sim q_0(\mathbf{z}_0) \tag{2}$$

Normalizing Flows

Apply a series of such mappings f_k, $k \in 1, \ldots, K$ with $K \in \mathbb{N}_+$ and obtain a **normalizing flow**.

$$\mathbf{x} = \mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0), \quad \mathbf{z}_0 \sim q_0(\mathbf{z}_0) \tag{2}$$

$$\mathbf{z}_K \sim q_K(\mathbf{z}_K) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1} \tag{3}$$

Normalizing Flows

Apply a series of such mappings f_k, $k \in 1, \ldots, K$ with $K \in \mathbb{N}_+$ and obtain a **normalizing flow**.

$$\mathbf{x} = \mathbf{z}_K = f_K \circ \cdots \circ f_1(\mathbf{z}_0), \quad \mathbf{z}_0 \sim q_0(\mathbf{z}_0) \tag{2}$$

$$\mathbf{z}_K \sim q_K(\mathbf{z}_K) = q_0(\mathbf{z}_0) \prod_{k=1}^{K} \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|^{-1} \tag{3}$$

$$\log q(\mathbf{z}_K) = \log q(\mathbf{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right| \tag{4}$$

Normalizing Flows

Loss Function :[1]

$$\begin{aligned}
\mathcal{F}(x) &= E_{\mathbf{z}\sim q}[\log q(\mathbf{z}) - \log P(x, \mathbf{z})] && (5)\\
&= E_{z\sim q}[\log q(\mathbf{z_K}) - \log P(x, \mathbf{z_K})] && (6)\\
&= E_{z\sim q}[\log q(\mathbf{z_0}) - \sum_{k=1}^{K} \log \left| \det \frac{\partial f_k}{\partial \mathbf{z_{k-1}}} \right| - \log P(x, \mathbf{z_K})] && (7)
\end{aligned}$$

[1] For the derivation of the Variational Free Energy, see Slide 26: Appendix

Planar Flows

- Transformation function f_k should satisfy two properties:
  - It is easily invertible.
  - Its Jacobian determinant is easy to compute.
- Planar Flows : $f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$

$$\psi(z) = h'(w^T z + b)w \tag{8}$$

$$\left|\det \frac{\partial f}{\partial z}\right| = \left|\det(I + u\psi(z)^T)\right| = \left|1 + u^T\psi(z)\right| \tag{9}$$

- Matrix-determinant lemma : $\det(I + uv^T) = (1 + v^T u)$

Normalizing Flows

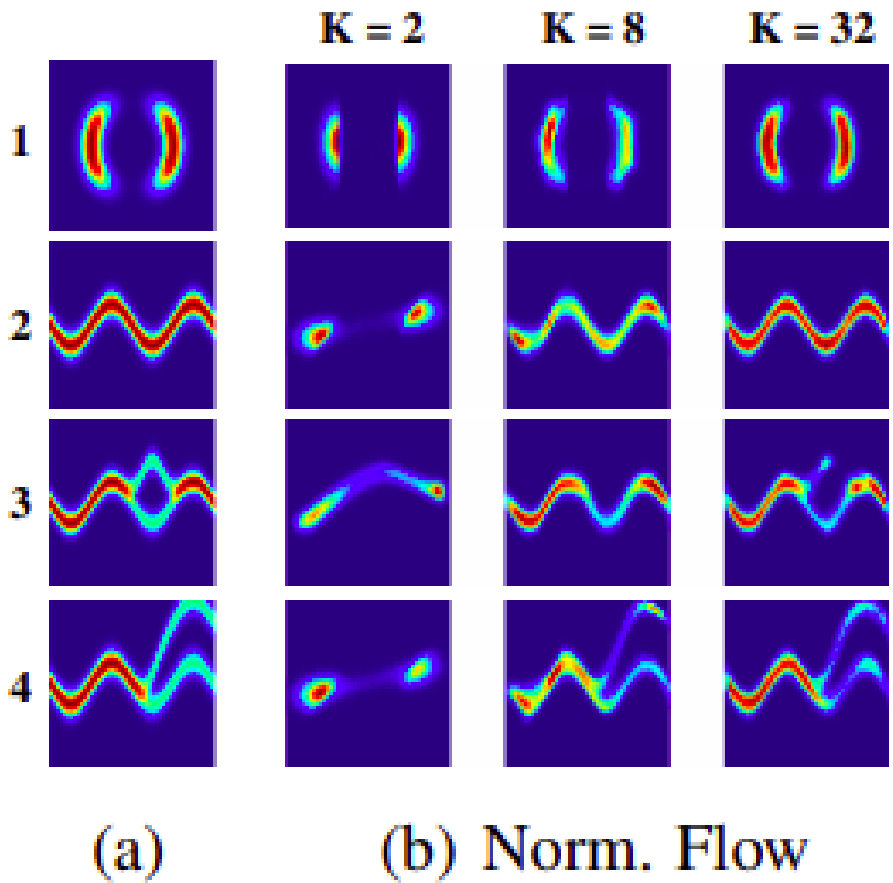


Figure: (a) Target Distribution. (b)) Approx posterior using NF[2]

[2] Image Source : Rezende and Mohamed (2015)

Autoregressive Models

- Autoregressive constraint : each output only depends on the data observed in the past, but not on the future ones.

Autoregressive Models

- Autoregressive constraint : each output only depends on the data observed in the past, but not on the future ones.

$$p(\mathbf{x}) = \prod_{i=1}^{D} p(x_i|x_1, \ldots, x_{i-1}) = \prod_{i=1}^{D} p(x_i|x_{1:i-1}) \tag{10}$$

Autoregressive Models

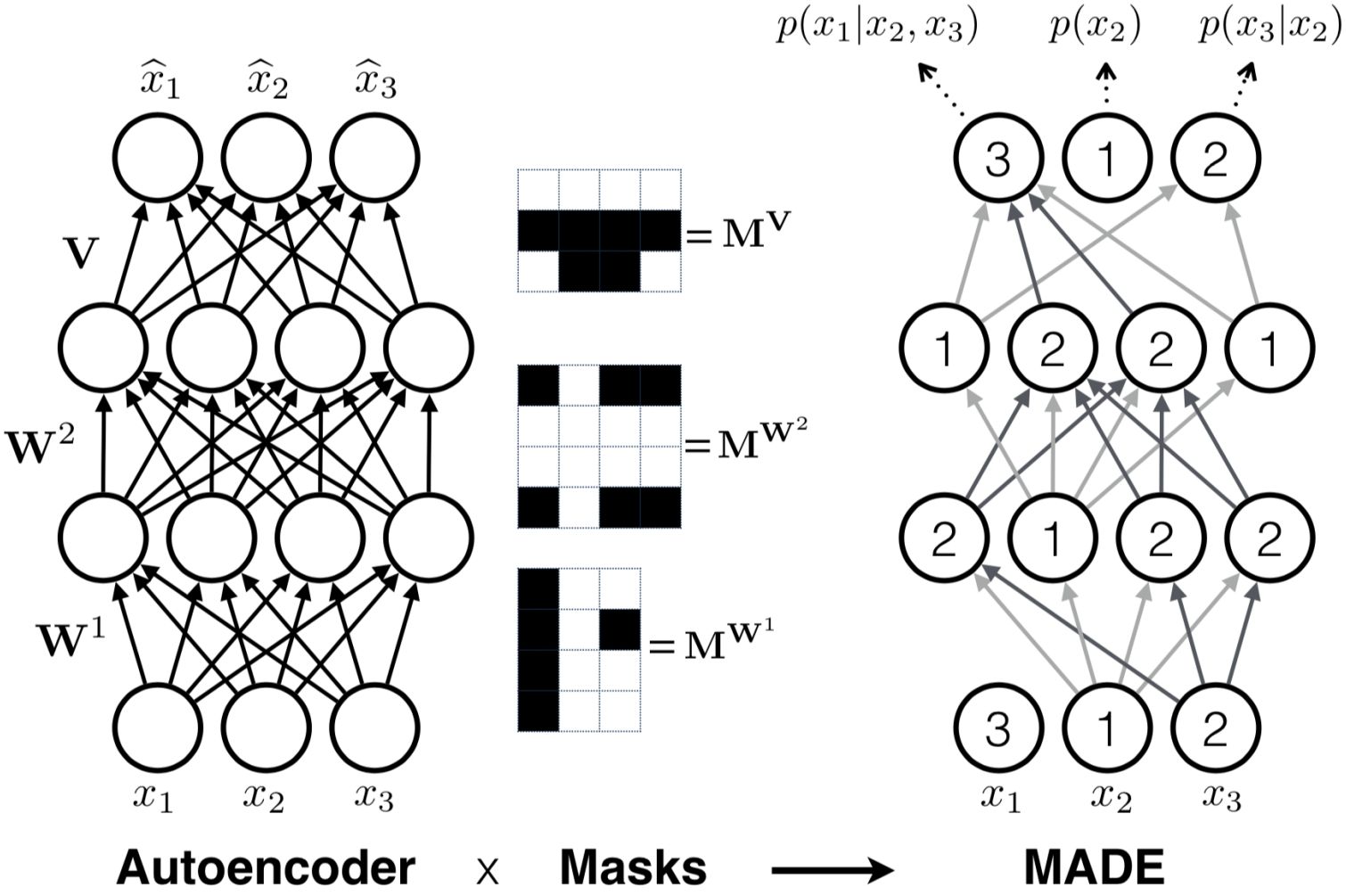


Figure: Demonstration MADE in a 3-layer NN. Germain et al. (2015)[3]

[3] Image Source : Germain et al. (2015)

Autoregressive Flows

- flow transformation framed as an autoregressive model — **autoregressive flow**.

Autoregressive Flows

- flow transformation framed as an autoregressive model — **autoregressive flow**.
- Autoregressive functions do have tractable determinant Jacobians.

Autoregressive Flows

- flow transformation framed as an autoregressive model — **autoregressive flow**.
- Autoregressive functions do have tractable determinant Jacobians.
- Let $f(\mathbf{z}) = \mathbf{z}'$. An autoregressive function would look like this:

$$z'_i = f(z_{1:i}) \quad \forall i \tag{11}$$

Autoregressive Flows

- flow transformation framed as an autoregressive model — **autoregressive flow**.
- Autoregressive functions do have tractable determinant Jacobians.
- Let $f(\mathbf{z}) = \mathbf{z}'$. An autoregressive function would look like this:

$$z'_i = f(z_{1:i}) \quad \forall i \tag{11}$$

- The Jacobian $J = \frac{\partial z'}{\partial z}$ becomes lower triangular.

$$\det J = \prod_{i=1}^{D} J_{ii} \tag{12}$$

Autoregressive Transformation

- Let $z' = f(z)$, and the first dimension of the transformed variable be defined by:

$$z \sim \mathcal{N}(0, I) \quad \text{[base distribution: } q(\mathbf{z})\text{]} \tag{13}$$

$$z'_0 = \mu_0 + \sigma_0 \odot z_0 \quad \text{[Randomly Sampled]} \tag{14}$$

Autoregressive Transformation

- Let $z' = f(z)$, and the first dimension of the transformed variable be defined by:

$$z \sim \mathcal{N}(0, I) \quad \text{[base distribution: } q(\mathbf{z})\text{]} \tag{13}$$

$$z'_0 = \mu_0 + \sigma_0 \odot z_0 \quad \text{[Randomly Sampled]} \tag{14}$$

- The other dimensions $i > 0$ are computed by :

$$z'_i = \mu_i(z'_{1:i-1}) + \sigma_i(z'_{1:i-1}) \cdot z_i \tag{15}$$

Inverse Autoregressive Transformation

- Inverse of Equation (14) and (15) are :

$$z_0 = \frac{z'_0 - \mu_0}{\sigma_0} \tag{16}$$

$$z_i = \frac{z'_i - \mu(z'_{1:i-1})}{\sigma(z'_{1:i-1})} \tag{17}$$

Inverse Autoregressive Transformation

- Inverse of Equation (14) and (15) are :

$$z_0 = \frac{z'_0 - \mu_0}{\sigma_0} \tag{16}$$

$$z_i = \frac{z'_i - \mu(z'_{1:i-1})}{\sigma(z'_{1:i-1})} \tag{17}$$

- $\mu(\cdot)$ and $\sigma(\cdot)$ are independent on z'_i, $\therefore$ we have a tractable determinant Jacobian

Inverse Autoregressive Transformation

- Inverse of Equation (14) and (15) are :

$$z_0 = \frac{z'_0 - \mu_0}{\sigma_0} \tag{16}$$

$$z_i = \frac{z'_i - \mu(z'_{1:i-1})}{\sigma(z'_{1:i-1})} \tag{17}$$

- $\mu(\cdot)$ and $\sigma(\cdot)$ are independent on z'_i, $\therefore$ we have a tractable determinant Jacobian

$$\det \left| \frac{d\mathbf{z}}{d\mathbf{z}'} \right| = \prod_{i=1}^{D} \frac{1}{\sigma_i(z_{1:i-1})} \tag{18}$$

$$\log \det \left| \frac{d\mathbf{z}}{d\mathbf{z}'} \right| = \sum_{i=1}^{D} -\log \sigma_i(z_{1:i-1}) \tag{19}$$

Inverse Autoregressive Flow (IAF)

- Kingma et al. (2016) has defined two functions $\mathbf{s}_t = \frac{1}{\sigma_t(\cdot)}$ and $\mathbf{m}_t = \frac{-\mu_t(\cdot)}{\sigma_t(\cdot)}$.
- $\mathbf{s}_t$ and $\mathbf{m}_t$ can be modeled by **autoregressive neural networks**.

$$
\begin{aligned}
z_t &= \frac{z_{t-1} - \mu_t(z_{t-1})}{\sigma_t(z_{t-1})} && (20)\\
&= \frac{z_{t-1}}{\sigma_t(z_{t-1})} - \frac{\mu_t(z_{t-1}))}{\sigma_t(z_{t-1})} && (21)\\
&= z_{t-1} \odot \mathbf{s}_t(z_{t-1}) + \mathbf{m}_t(z_{t-1}) && (22)
\end{aligned}
$$

Inverse Autoregressive Flow (IAF)

- Kingma et al. (2016) has defined two functions $\mathbf{s}_t = \frac{1}{\sigma_t(\cdot)}$ and $\mathbf{m}_t = \frac{-\mu_t(\cdot)}{\sigma_t(\cdot)}$.
- $\mathbf{s}_t$ and $\mathbf{m}_t$ can be modeled by **autoregressive neural networks**.

$$
\begin{aligned}
z_t &= \frac{z_{t-1} - \mu_t(z_{t-1})}{\sigma_t(z_{t-1})} && (20)\\
&= \frac{z_{t-1}}{\sigma_t(z_{t-1})} - \frac{\mu_t(z_{t-1}))}{\sigma_t(z_{t-1})} && (21)\\
&= z_{t-1} \odot \mathbf{s}_t(z_{t-1}) + \mathbf{m}_t(z_{t-1}) && (22)
\end{aligned}
$$

- And finally for numerical stability they modify Equation (22)

$$
\begin{aligned}
g_t &= \text{sigmoid}(\mathbf{s}_t(z_{t-1})) && (23)\\
z_t &= z_{t-1} \odot g_t + (1 - g_t) \odot \mathbf{m}_t(z_{t-1}) && (24)
\end{aligned}
$$

Neural Autoregressive Flow

- Re-write Equation (24) as a combination of an autoregressive **conditioner**, c and an invertible **transformer**, τ

$$y \doteq f(x_{1:t}) = \tau(c(x_{1:t-1}), x_t) \tag{25}$$

Neural Autoregressive Flow

- Re-write Equation (24) as a combination of an autoregressive **conditioner**, c and an invertible **transformer**, τ

$$y \doteq f(x_{1:t}) = \tau(c(x_{1:t-1}), x_t) \tag{25}$$

- c can be computed by an autoregressive model such as **MADE**(Germain et al. (2015)).

Neural Autoregressive Flow

- Re-write Equation (24) as a combination of an autoregressive **conditioner**, c and an invertible **transformer**, τ

$$y \doteq f(x_{1:t}) = \tau(c(x_{1:t-1}), x_t) \tag{25}$$

- c can be computed by an autoregressive model such as **MADE**(Germain et al. (2015)).
- The transformation function, τ as given by Kingma et al. (2016) is:

$$\tau(\mu, \sigma, x_t) = \sigma x_t + (1 - \sigma)\mu \tag{26}$$

Neural Autoregressive Flow

- Re-write Equation (24) as a combination of an autoregressive **conditioner**, c and an invertible **transformer**, τ

$$y \doteq f(x_{1:t}) = \tau(c(x_{1:t-1}), x_t) \tag{25}$$

- c can be computed by an autoregressive model such as **MADE**(Germain et al. (2015)).
- The transformation function, τ as given by Kingma et al. (2016) is:

$$\tau(\mu, \sigma, x_t) = \sigma x_t + (1 - \sigma)\mu \tag{26}$$

Notice the similarity between Eq (24) and (26).

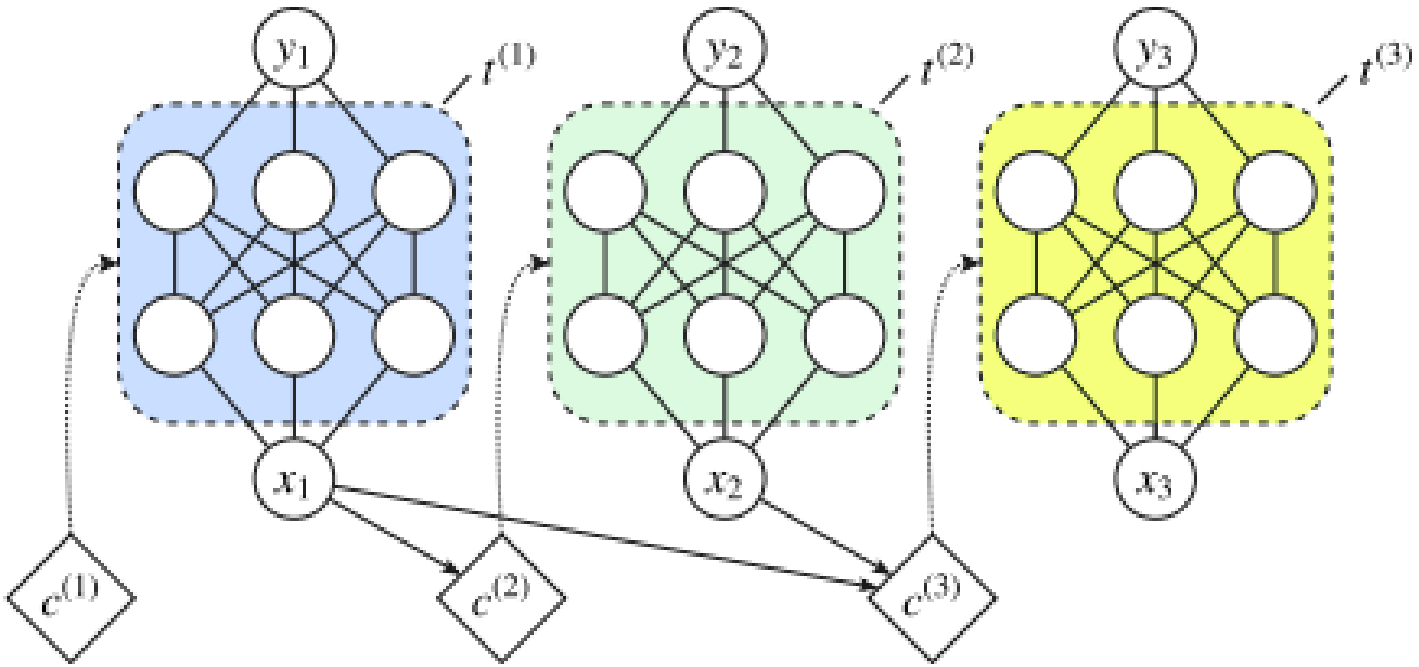
Neural Autoregressive Flow

- Two requirements for transformer τ
  - must be invertible as a function of x_t
  - $\frac{dy_t}{dx_t}$ must be cheap to compute.
- In IAF, τ is affine, trivially invertible. In NAF, τ is a neural network.

$$\tau(c(x_{1:t-1}), x_t) = DNN(x_t; \phi = c(x_{1:t-1})) \tag{27}$$

- where ϕ are the **pseudo-parameters**

Architecture of NAF



(a) NAF: each $c^{(i)}$ is a neural network that predicts pseudo-parameters for $t^{(i)}$, which in turns processes x_i.

Figure: Block Diagram of NAF Architecture.Here $t^{(i)}$ means $\tau^{(i)}$[4]

[4]Image Source : De Cao et al. (2019)

Architecture of NAF

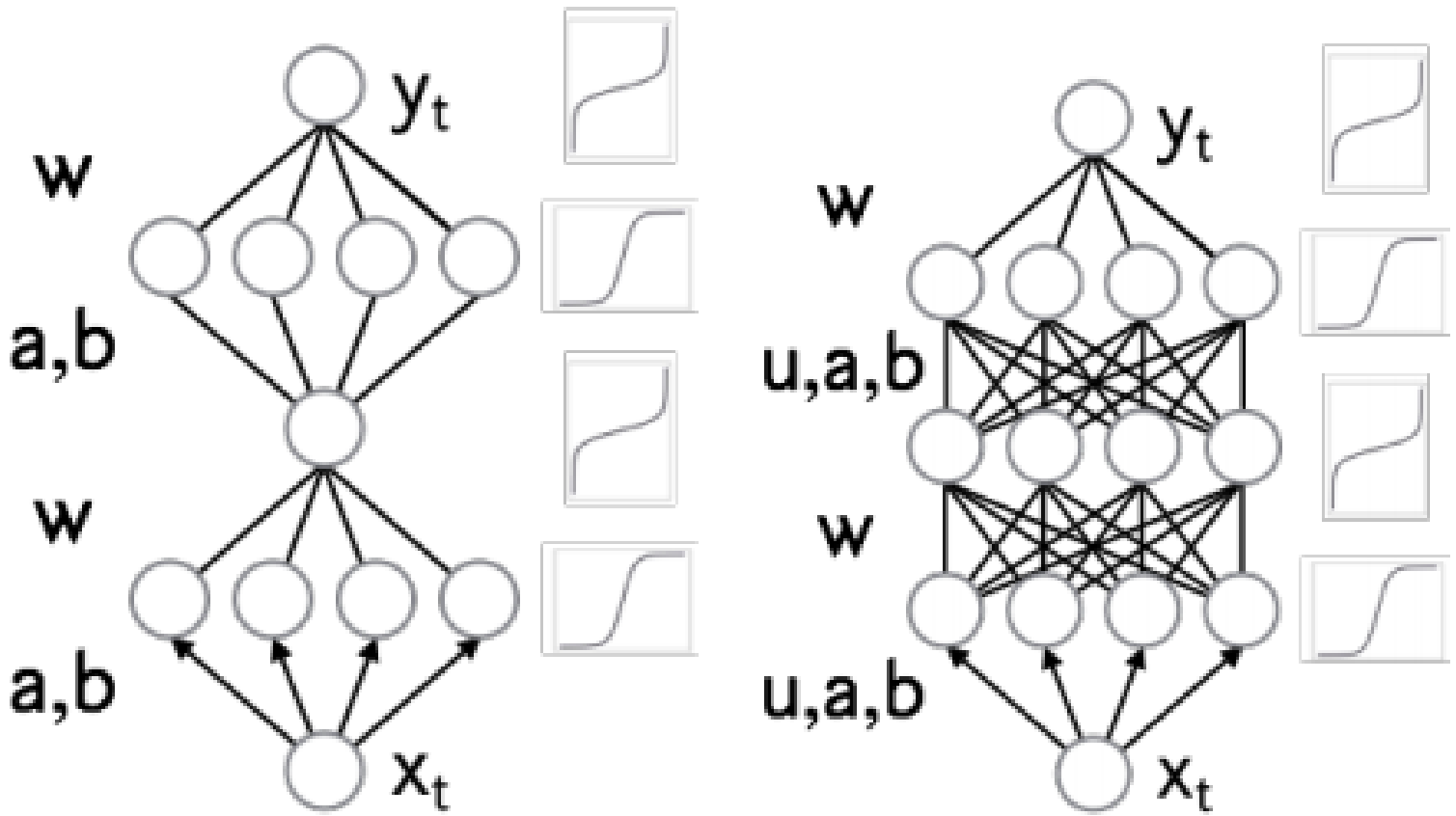


Figure: Proposed Transformer Networks (τ). **Left**: Deep Sigmoidal Flows(DSF). **Right**: Deep Dense Sigmoidal Flows(DDSF).[5]

[5] Image Source: Huang et al. (2018)

More on DSF and DDSF

- DSF:

$$y_t = \sigma^{-1}(\underbrace{w^T}_{1\times d} \cdot \sigma(\underbrace{a}_{d\times 1} \cdot \underbrace{x_t}_{1\times 1} + \underbrace{b}_{d\times 1})) \tag{28}$$

where $0 < w_{i,h} < 1, \sum_i w_{i,j} = 1$ and d denotes the number of hidden units.

More on DSF and DDSF

- DSF:

$$y_t = \sigma^{-1}(\underbrace{w^T}_{1\times d} \cdot \sigma(\underbrace{a}_{d\times 1} \cdot \underbrace{x_t}_{1\times 1} + \underbrace{b}_{d\times 1})) \tag{28}$$

where $0 < w_{i,h} < 1, \sum_i w_{i,j} = 1$ and d denotes the number of hidden units.

- DDSF :

$$h^{(l+1)} = \sigma^{-1}(\underbrace{w^{(l+1)}}_{d_{l+1}\times d_{l+1}} \cdot \sigma(\underbrace{a^{(l+1)}}_{d_{l+1}} \odot \underbrace{u^{(l+1)}}_{d_{l+1}\times d_l} \cdot \underbrace{h^{(l)}}_{d_l} + \underbrace{b^{(l+1)}}_{d_{l+1}})) \tag{29}$$

for $1 < l < L$, $h_0 = x$ and $y = h_L; d_0 = d_L = 1$; $\mathbf{w}$ and $\mathbf{u}$ are softmax-weighted.

More on DSF and DDSF

- DSF:

$$y_t = \sigma^{-1}(\underbrace{w^T}_{1\times d} \cdot \sigma(\underbrace{a}_{d\times 1} \cdot \underbrace{x_t}_{1\times 1} + \underbrace{b}_{d\times 1})) \tag{28}$$

where $0 < w_{i,h} < 1, \sum_i w_{i,j} = 1$ and d denotes the number of hidden units.

- DDSF :

$$h^{(l+1)} = \sigma^{-1}(\underbrace{w^{(l+1)}}_{d_{l+1}\times d_{l+1}} \cdot \sigma(\underbrace{a^{(l+1)}}_{d_{l+1}} \odot \underbrace{u^{(l+1)}}_{d_{l+1}\times d_l} \cdot \underbrace{h^{(l)}}_{d_l} + \underbrace{b^{(l+1)}}_{d_{l+1}})) \tag{29}$$

for $1 < l < L$, $h_0 = x$ and $y = h_L$;$d_0 = d_L = 1$; **w** and **u** are softmax-weighted.

- Log-determinant of the Jacobian:

$$\nabla_x y = \left[\nabla_{h^{(L-1)}} h^{(L)}\right] \left[\nabla_{h^{(L-2)}} h^{(L-1)}\right], \cdots, \left[\nabla_{h^{(0)}} h^{(1)}\right] \tag{30}$$

Result

Figure: **Left**: Target Distribution, **Right**: Modeled by NAF.[6]
Resemblance of the distribution with any animal, living or extinct, is purely co-incidental.

[6] Image Source: https://medium.com/element-ai-research-lab/neural-autoregressive-flows-f164d6b8e462

Comparison with AAF(IAF)

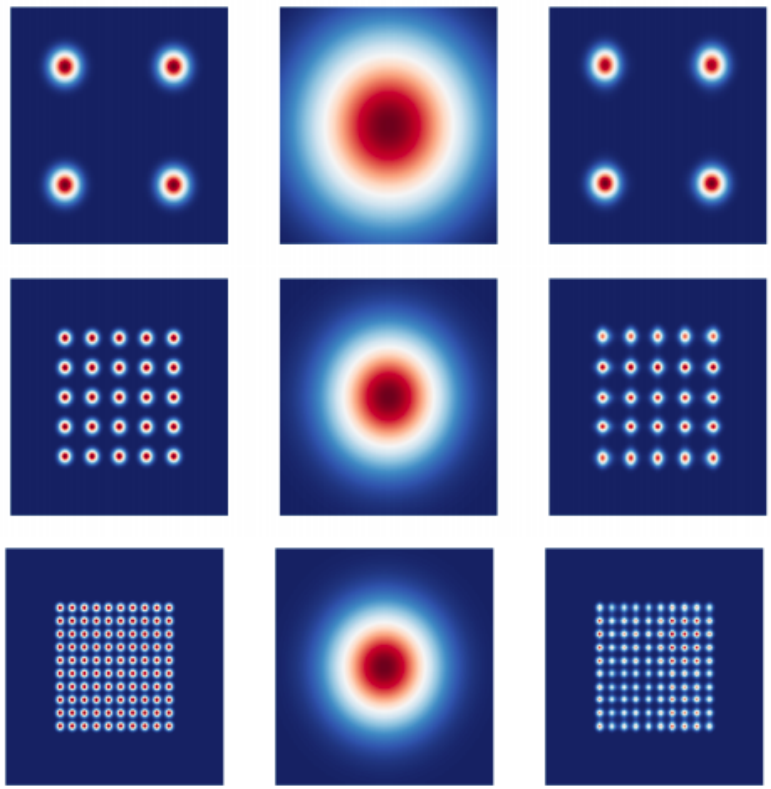


Figure: Fitting grid of Gaussian distributions using maximum likelihood. Left: true distribution. Center: AAF. Right: NAF[7]

[7] Image Source : Huang et al. (2018)

Detailed Comparison

Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
MADE MoG	0.40 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
MAF-affine (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	-11.75 ± 0.44	155.69 ± 0.28
MAF-affine (10)	0.24 ± 0.01	10.08 ± 0.02	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF-affine MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
TAN (various architectures)	0.48 ± 0.01	11.19 ± 0.02	-15.12 ± 0.02	-11.01 ± 0.48	157.03 ± 0.07
MAF-DDSF (5)	$\mathbf{0.62 \pm 0.01}$	11.91 ± 0.13	$\mathbf{-15.09 \pm 0.40}$	$\mathbf{-8.86 \pm 0.15}$	$\mathbf{157.73 \pm 0.04}$
MAF-DDSF (10)	0.60 ± 0.02	$\mathbf{11.96 \pm 0.33}$	-15.32 ± 0.23	-9.01 ± 0.01	157.43 ± 0.30
MAF-DDSF (5) valid	0.63 ± 0.01	11.91 ± 0.13	15.10 ± 0.42	-8.38 ± 0.13	172.89 ± 0.04
MAF-DDSF (10) valid	0.60 ± 0.02	11.95 ± 0.33	15.34 ± 0.24	-8.50 ± 0.03	172.58 ± 0.32

Figure: Test log-likelihood and error bars of 2 standard deviations on the 5 datasets.[8]

[8] Table taken from : https://medium.com/element-ai-research-lab/neural-autoregressive-flows-f164d6b8e462

References I

De Cao, N., Titov, I., and Aziz, W. (2019). Block neural autoregressive flow. *arXiv preprint arXiv:1904.04676*.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. *arXiv preprint arXiv:1804.00779*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751.

Rezende, D. J. and Mohamed, S. (2015). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Acknowledgements

Slides made with help from the following blogs :

- https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html
- http://akosiorek.github.io/ml/2018/04/03/norm_flows.html
- https://medium.com/element-ai-research-lab/neural-autoregressive-flows-f164d6b8e462
- https://www.ritchievink.com/

Thank you

Appendix

- Derivation of the variational free enery :

$$
\begin{aligned}
F(x) &= D_{KL}(Q(z)||P(z)) - E_{z\sim Q}[\log P(x|z)] && (31)\\
&= \int_z Q(z) \log \frac{Q(z)}{P(z)} dz - E_{z\sim Q}[\log P(x|z)] && (32)\\
&= E_{z\sim Q}[\log \frac{Q(z)}{P(z)}] - E_{z\sim Q}[\log P(x|z)] && (33)\\
&= E_{z\sim Q}[\log \frac{Q(z)}{P(z)} - \log P(x|z)] && (34)\\
&= E_{z\sim Q}[\log Q(z) - \log P(z) - \log P(x|z)] && (35)\\
&= E_{z\sim Q}[\log Q(z) - (\log P(z) + \log P(x|z))] && (36)\\
&= E_{z\sim Q}[\log Q(z) - \log P(x,z))] && (37)
\end{aligned}
$$