Learning and Generalization in Overparameterized Normalizing Flows

Kulin Shah, Amit Deshpande, Navin Goyal

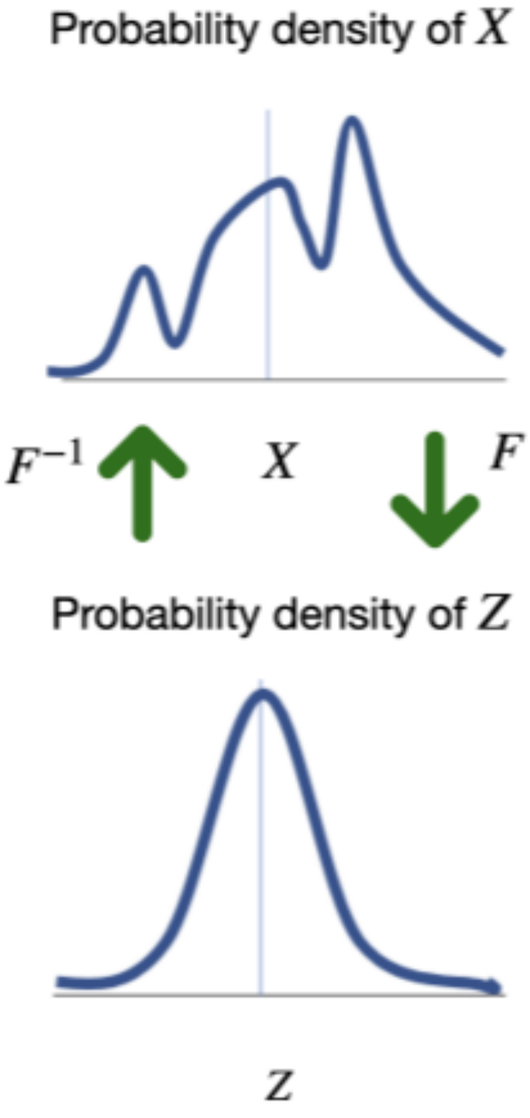
Yunshu Ouyang, April 13 2022

Generative models

- Goals of a generative model: generate new samples from the distribution and give a probability density estimate at any queried point
- Generative Adversarial Networks (GANs), Variational AutoEncoders (VAEs) fail at producing a probability density estimate for new data points.

Glow (normalizing flow)

(Autoregressive) Normalizing Flows (NFs)



Target data random variable whose distribution we want to learn

$$X \in \mathbb{R}^d$$

Distribution with finite support

WLOG $p_X(u) = 0 \forall ||u||_2 \geq 1$

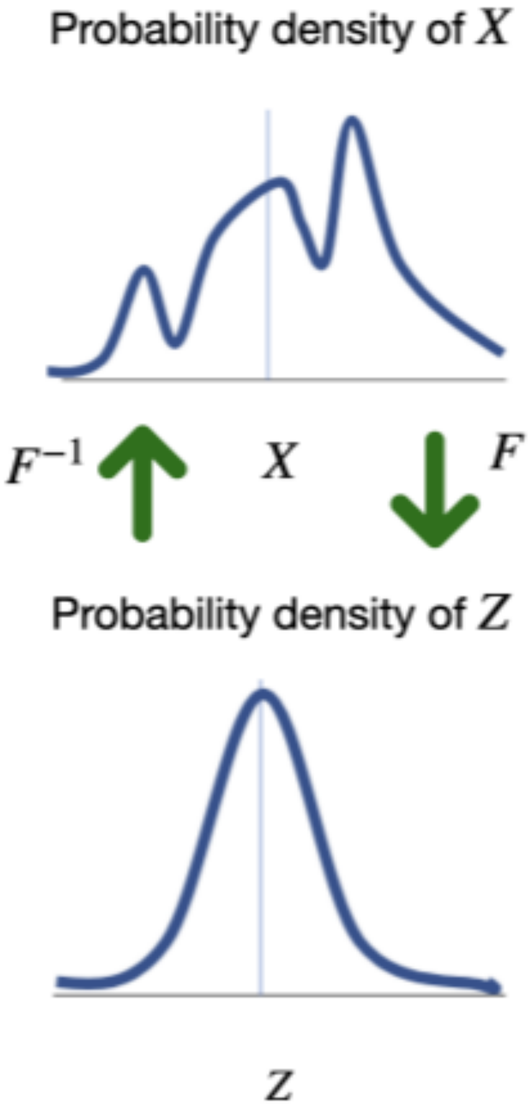
Base random variable such as standard Gaussian or exponential

$$Z \in \mathbb{R}^d$$

$$p_Z(z) = \begin{cases} \exp\left(-\sum_{i=1}^{d} z_i\right) & z_i \geq 0 \forall i \\ 0 & \text{otherwise} \end{cases}$$

Exponential distribution

(Autoregressive) Normalizing Flows (NFs)



Target data random variable whose distribution we want to learn

$$X \in \mathbb{R}^d$$

Base random variable such as standard Gaussian or exponential

$$Z \in \mathbb{R}^d$$

Goal: learn a differentiable invertible map

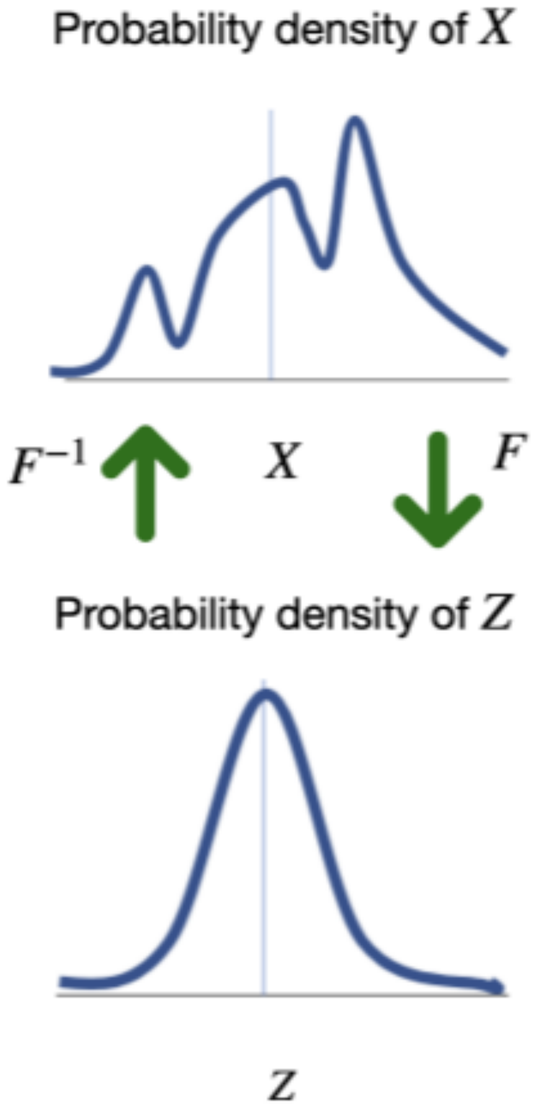
$$f_X : \mathbb{R}^d \to \mathbb{R}^d$$

Autoregressive

$$f_X(x) = \left(f_{X,1}(x_1), f_{X,2}(x_{1:2}), \ldots, f_{X,d}(x_{1:d})\right)$$

$f_{X,i}(x_{1:i})$ strictly monotonically increasing function in x_i for any fixed value of $x_{1:(i-1)}$

(Autoregressive) Normalizing Flows (NFs)



Density evaluation

$$p_{f,Z}(x) = p_Z(f(x)) \left| \det \left(\frac{\partial f(x)}{\partial x} \right) \right|$$

Maximum log-likelihood objective

$$\max_f \frac{1}{n} \sum_{x \in \mathcal{X}} \log p_{f,Z}(x)$$

$$= \max_f \frac{1}{n} \Big[\sum_{x \in \mathcal{X}} \log p_Z(f(x)) + \sum_{x \in \mathcal{X}} \log \Big(\det \Big(\frac{\partial f(x)}{\partial x} \Big) \Big) \Big]$$

Goal: Theoretical understanding of learning and generalization of autoregressive normalizing flows when they are parameterized using a neural network.

Contributions

- Theoretical and empirical evidence that for constrained NFs, overparameterization hurts the training.
- Unconstrained NFs can efficiently learn any reasonable data distribution when the underlying network is overparameterized.

Supervised learning analysis

Proof technique for analyzing training and generalization for one-hidden layer neural networks

$$x \in \mathbb{R}^d \sim D \longrightarrow y = h(x) \in \mathbb{R}$$

unknown function modeled by a one-hidden-layer neural network

its complexity is the size of the NN

Loss function

$$L_s(N^{(t)}, x) = (N(x; \theta^{(t)}) - y)^2$$

SGD to update the parameters

Problem: optimizing the square loss is a non-convex problem!

Instead, consider the pseudo-network, a linear approximation of the NN

$$P(x;\theta) = \sum_{r=1}^{m} \bar{a}_r(\rho(\langle \bar{w}_r, x\rangle + \bar{b}_r) + \rho'(\langle \bar{w}_r, x\rangle + \bar{b}_r)\left(\langle w_r, x\rangle + b_r\right))$$

Constrained Normalizing Flows

Represent the function directly using neural networks

$$f_X(x) = \left(f_{X,1}(x_1), f_{X,2}(x_{1:2}), \ldots, f_{X,d}(x_{1:d})\right)$$

d neural networks $N_1, \ldots N_d$

$f_{X,i}(x_{1:i}) = N_i(x_{1:i})$

Need to introduce constraints to make the NN monotone

$$N(x_{1:i}; \theta_i)$$

$$= \tau \sum_{r=1}^{m} \bar{a}_{i,r} \tanh\left(\langle \bar{w}_{i,r} + w_{i,r}, x_{1:i} \rangle + (\bar{b}_{i,r} + b_{i,r})\right)$$

$$\text{with constraints } \bar{w}_{i,r,i} + w_{i,r,i} \geq \epsilon, \text{ for all } r.$$

$\bar{a}_{i,r} \geq 0$ and $\bar{w}_{i,r} + w_{i,r} \geq 0 \; \forall r$

$\bar{a}_{i,r}, \bar{w}_{i,r}, \bar{b}_{i,r}$ fixed

$w_{i,r}, b_{i,r}$ training params

Optimization: Projected SGD

Constrained Normalizing Flows

Represent the function directly using neural networks

$$P(x_{1:i};\theta_i) = P_c(x_{1:i}) + P_\ell(x_{1:i};\theta_i)$$

with constraints $\bar{w}_{i,r,i} + w_{i,r,i} \geq \epsilon$ for all r, where

$$P_c(x_{1:i}) = \tau \sum_{r=1}^{m} \bar{a}_{i,r} \tanh(\langle \bar{w}_{i,r}, x_{1:i} \rangle + \bar{b}_{i,r}) \quad \text{and}$$

$$P_\ell(x_{1:i};\theta_i) = \tau \sum_{r=1}^{m} \bar{a}_{i,r} \tanh'(\langle \bar{w}_{i,r}, x_{1:i} \rangle + \bar{b}_{i,r}) \left(\langle w_{i,r}, x_{1:i} \rangle + b_{i,r}\right).$$

Constrained Normalizing Flows

Represent the function directly using neural networks

Theorem 3.1. *For any $\epsilon > 0$, for any $i \in [d]$, any hidden layer size $m \geq \Omega\big(\mathrm{poly}\left(C(F^*), \frac{1}{\epsilon}\right)\big)$, by choosing learning rate $\eta = O\big(\frac{\epsilon}{m\tau\epsilon_a^2 \log m}\big)$ and $T = O\big(\frac{C(F^*)}{\epsilon^2}\big)$, with at least probability 0.9, there exist constants $\alpha_i \in \mathbb{R}^i$ and $\beta \in \mathbb{R}$ for which projected SGD after T iterations gives*

$$|N(x_{1:i}; \theta_i^{(T)}) - (\langle \alpha_i, x_{1:i} \rangle + \beta)| \leq O(\epsilon), \qquad (3.1)$$

for all x with $\|x\|_2 \leq 1$.

Unconstrained Normalizing Flows

Represent the Jacobean matrix using neural networks

Model diagonal entries of the Jacobian by NNs

$$\frac{\partial f_i(x_{1:i})}{\partial x_i} = \phi(N(x_{1:i}; \theta_i)),$$

where ϕ is ELU $+ 1$ function given by

$$\phi(u) = e^u \, \mathbb{I}\left[u < 0\right] + (u + 1) \, \mathbb{I}\left[u \geq 0\right] \text{ and}$$

$$N(x_{1:i}; \theta_i) = \sum_{r=1}^{m} \bar{a}_{i,r} \, \rho(\langle \bar{w}_{i,r} + w_{i,r}, \tilde{x}_{1:i} \rangle + \left(\bar{b}_{i,r} + b_{i,r}\right))$$

with $\rho =$ ReLU.

Reconstruct the function by integration

$$f_i(x_{1:i}) = \int_{-1}^{x_i} \frac{\partial f_i(x_1, x_2, \ldots, x_{i-1}, t)}{\partial t} \mathrm{d}t$$

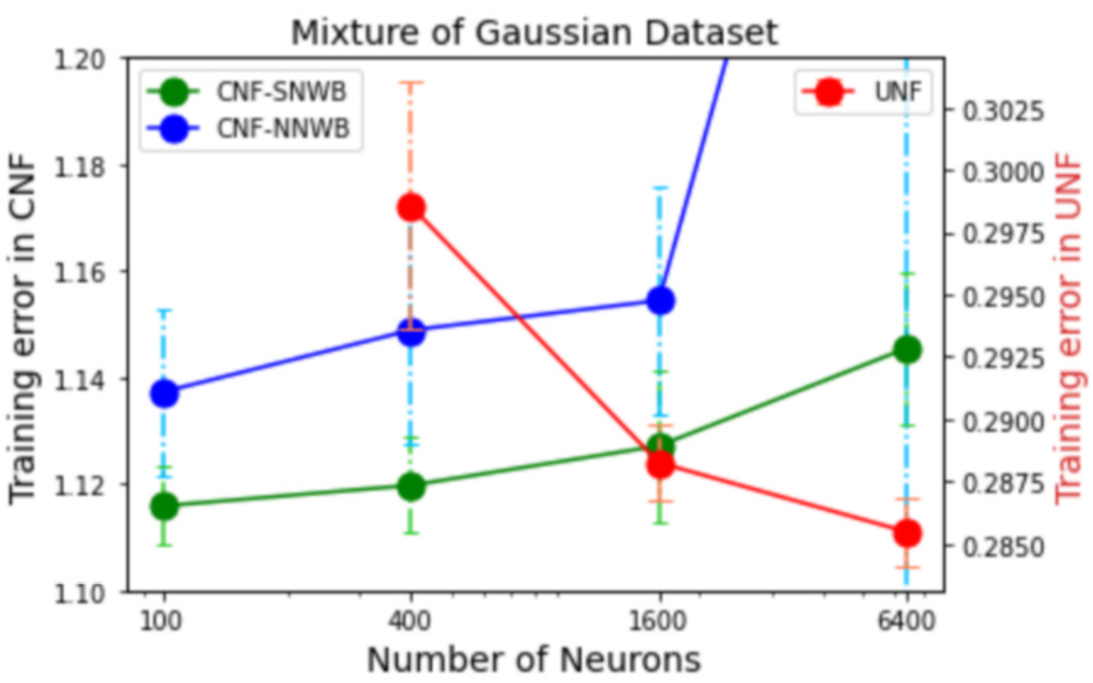
Unconstrained Normalizing Flows

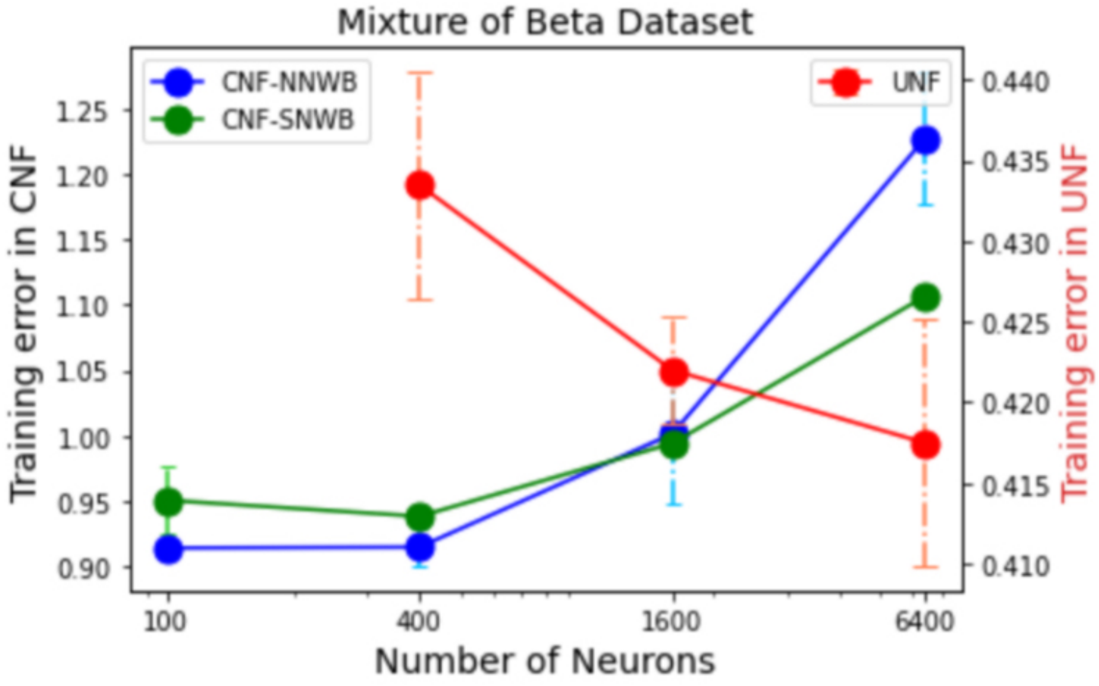
Represent the Jacobean matrix using neural networks

Theorem 4.1. *For any $\epsilon > 0$ and for any target function F^* with finite $\frac{\partial F_i^*(x_{1:i})}{\partial x_i}$ for all $i \in [d]$, hidden layer size $m \geq \frac{C(F^*,\epsilon)}{\epsilon^2}$, the number of samples $n \geq \frac{C(F^*,\epsilon)}{\epsilon^2}$, the number of quadrature points $Q \geq O(\frac{C(F^*,\epsilon)}{\epsilon})$ and total time steps $T \geq O(\frac{C(F^*,\epsilon)}{\epsilon^2})$ with probability at least 0.9, we have*

$$\mathbb{E}_{\text{sgd}}\left[\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}_{x\sim\mathcal{D}}L(f^{(t)},x)\right] - \mathbb{E}_{x\sim\mathcal{D}}\left[L(F^*,x)\right] = O(\epsilon)$$

Experimental results





Since results in supervised learning suggest that overparaterization makes training faster, the results on CNF are novel and surprising.

Conclusion and limitations

- First end-to-end theoretical analysis of normalizing flows
- For the analysis the NF architecture is distilled to essentials, the practical architectures are far more elaborate and performant.
- The paper only considers autoregressive structure of the flow models. Many other structures exist, ie. coupling flows, residual flows.
- In practice, flow models are sequentially composed to learn more flexible target distributions. Composition of invertible flow models not yet analyzed.