Normalizing Flows

Mauro Camara Escudero

School of Mathematics, University of Bristol

Table of contents

Normalizing Flows at a Glance

- **What are they?** Normalizing Flows define expressive probability distributions that we can sample and evaluate.
- **How do we learn them?** We specify transformations in advance and learn their parameters.
- **How?** Transforming a simple density into a more complex one via a chain of invertible transformations.
- **How expressive are they?** If the target distribution $p^*_{\mathbf{z}}(\mathbf{z})$ satisfies $p^*_{\mathbf{z}}(\mathbf{z}) > 0$ for all $\mathbf{z}$ and $\mathbb{P}(z'_i \leq z_i \mid \mathbf{z}_{<i})$ are differentiable w.r.t. $(z_i, \mathbf{z}_{<i})$, then it can be represented by starting with an initial uniform distribution, and the Jacobian of the transformation is lower triangular.
- **What is the challenge?** Finding transformations with a lower-triangular Jacobian that do not restrict the expressive power of the distribution.

Background

Variational Auto-Encoders Review

- Recall VAEs are used for latent posterior inference, parameter estimation, and generative modelling.
- Optimize Evidence Lower Bound (ELBO) using SGD.

$$\mathcal{L}_{\theta,\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})\right]$$

- This requires computing gradients with respect to ϕ

$$\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\theta,\boldsymbol{\phi}}(\mathbf{x}) = \nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\cdot\right]$$

- To use Monte Carlo estimates we reparametrized $q_{\boldsymbol{\phi}}$ to exchange $\nabla_{\boldsymbol{\phi}}$ and $\mathbb{E}$. We call this the reparametrization trick.

$$\left.\begin{aligned} \boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) \quad \text{independent of } \phi \\ \mathbf{z} &= g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon},\mathbf{x}) \quad \text{deterministic} \end{aligned}\right\} \implies \quad \mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$$

Reparametrization Trick: Diffeomorphism of a Simple Distribution

- Given a random variable $\boldsymbol{\epsilon}$ with distribution $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$, a transformed random variable $\mathbf{z} = \mathbf{g}(\boldsymbol{\epsilon})$ has distribution

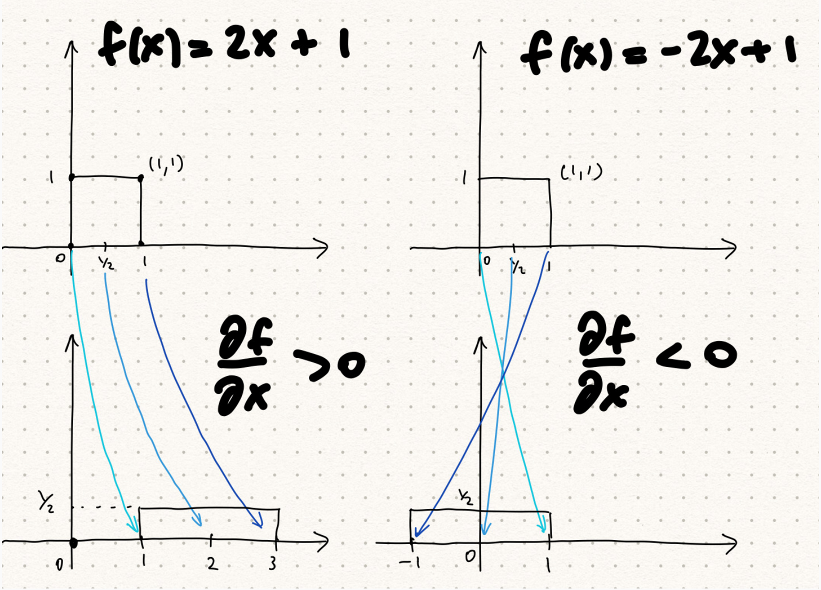
$$\begin{aligned} p_{\mathbf{z}}(\mathbf{z}) &= p_{\boldsymbol{\epsilon}}(\mathbf{g}^{-1}(\mathbf{z}))|\det J_{\mathbf{g}}(\mathbf{g}^{-1}(\mathbf{z}))|^{-1} \\ &= p_{\boldsymbol{\epsilon}}(\mathbf{g}^{-1}(\mathbf{z}))|\det J_{\mathbf{g}^{-1}}(\mathbf{z})| \end{aligned}$$

  where the Jacobian is given by

$$J_{\mathbf{g}^{-1}}(\mathbf{z}) = \begin{pmatrix} \frac{\partial g_1^{-1}}{\partial \epsilon_1} & \cdots & \frac{\partial g_D^{-1}}{\partial \epsilon_D} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_D^{-1}}{\partial \epsilon_1} & \cdots & \frac{\partial g_D^{-1}}{\partial \epsilon_D} \end{pmatrix}$$

- For this to work we need $\mathbf{g}$ to be a diffeomorphism: g^{-1} exists and both g and g^{-1} are differentiable.

Change of Variable, Change of Volume



Normalizing Flows

Composition of Diffeomorphisms

- Normalizing Flows generalize the reparametrization trick so that the resulting distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is more expressive.
- Suppose we have K diffeomorphisms $\mathbf{g}_k$. These are composable and their composition is a diffeomorphism.
- Define $\boldsymbol{\epsilon}_k = \mathbf{g}_k(\boldsymbol{\epsilon}_{k-1})$ for $k = 1, \ldots, K$ with $\boldsymbol{\epsilon}_0 = \boldsymbol{\epsilon}$ and $\boldsymbol{\epsilon}_K = \mathbf{z}$.
- Then their composition $\mathbf{g} = \mathbf{g}_K \circ \cdots \circ \mathbf{g}_1$ is invertible

$$\mathbf{z} = (\mathbf{g}_K \circ \cdots \circ \mathbf{g}_1)(\boldsymbol{\epsilon}) \quad \text{with} \quad \boldsymbol{\epsilon} = (\mathbf{g}_1^{-1} \circ \cdots \mathbf{g}_K^{-1})(\mathbf{z})$$

- And differentiable

$$\det J_{\mathbf{g}_K \circ \cdots \circ \mathbf{g}_1}(\boldsymbol{\epsilon}) = \prod_{k=1}^{K} \det J_{\mathbf{g}_k}(\boldsymbol{\epsilon}_{k-1})$$

Normalizing Flows: Sampling and Evaluating

- Idea: By composing simple diffeomorphisms $\mathbf{g}_k$ together, we can transform a base distribution $p_{\boldsymbol{\epsilon}}$ into a more complex one $p_{\mathbf{z}}$.
- Sampling:
  - Sample from base distribution: $\boldsymbol{\epsilon}^{(1)}, \ldots, \boldsymbol{\epsilon}^{(N)} \overset{\text{i.i.d.}}{\sim} p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$
  - Feed through the flow: $\mathbf{z}^{(i)} = \mathbf{g}(\boldsymbol{\epsilon}^{(i)})$ for $i = 1, \ldots, N$
- Evaluation: Using $\boldsymbol{\epsilon}_k = \mathbf{g}_{k+1}^{-1} \circ \cdots \circ \mathbf{g}_K^{-1}(\mathbf{z})$

$$\ln p_{\mathbf{z}}(\mathbf{z}) = \ln p_{\boldsymbol{\epsilon}}(\mathbf{g}^{-1}(\mathbf{z})) + \sum_{k=1}^{K} \ln \left| \det J_{\mathbf{g}_k^{-1}}(\boldsymbol{\epsilon}_k) \right|$$

- Sampling efficiency depends on $\sim p_{\boldsymbol{\epsilon}}$ and $\mathbf{g}(\cdot)$.
- Evaluating efficiency depends on $\mathbf{g}^{-1}(\cdot)$, $p_{\boldsymbol{\epsilon}}(\cdot)$ and log-det-jacobian.

What the area of Normalizing Flows is all about

- Depending on application one has to decide whether sampling or evaluating efficiency is more important, as this is a trade-off.
- One aims to design Normalizing Flows optimizing sampling or evaluating efficiency. This means choosing which of the operations above are efficient.
- By choosing to model $\mathbf{g}^{-1}$ rather than $\mathbf{g}$ as normalizing flow, one obtains an ”inverse” flow, with opposite properties (see MAF and IAF).

Universality and Duality

Universality

By starting with a base distribution that is uniformly distributed in $(0,1)^D$, we can represent any distribution satisfying:

- $p_{\mathbf{z}}(\mathbf{z}) > 0$ for all $\mathbf{z}$
- $\mathbb{P}(z_i' \leq z_i \mid \mathbf{z}_{<i})$ are differentiable wrt $(z_i, \mathbf{z}_{<i})$

Proof sketch:

- $p_{\mathbf{z}}(\mathbf{z}) = \prod_{i=1}^{D} p_{\mathbf{z}}(z_i \mid \mathbf{z}_{<i}) > 0$
- Want to show $p_{\mathbf{z}}(z_i \mid \mathbf{z}_{<i}) = \frac{\partial G_i}{\partial z_i}$ where G_i is the element-wise application of a function $\mathbf{G} : \mathbf{z} \to \boldsymbol{\epsilon}$
- Then as long as $\mathbf{G}$ has a lower-triangular Jacobian, then $p_{\mathbf{z}}(\mathbf{z}) = \det J_{\mathbf{G}}(\mathbf{z}) > 0$ and by the inverse function theorem $\mathbf{G}$ is invertible and it's inverse is also differentiable.

This can be proved by defining each G_i as a one of the conditional CDFs.

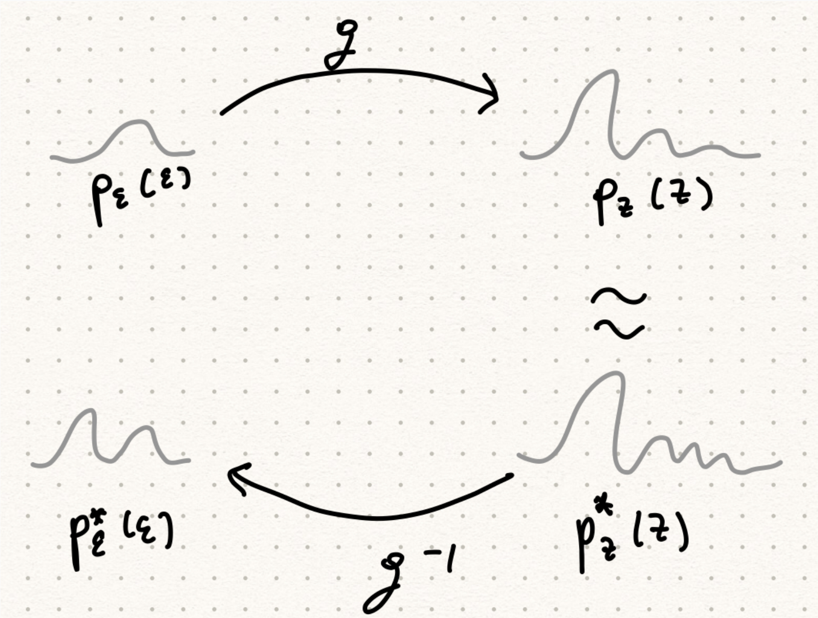
KL divergence Duality

Recall $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ is transformed to $p_{\mathbf{z}}(\mathbf{z})$ via $\mathbf{g}(\cdot)$, hoping to be close to the target $p^*_{\mathbf{z}}(\mathbf{z})$. One can also consider $p^*_{\mathbf{z}}(\mathbf{z})$ as the base, $\mathbf{g}^{-1}$ as the flow and thus $p^*_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ as the distribution that the training data would follow if passed through it.

- Maximum Likelihood: Minimize $\mathrm{KL}(p^*_{\mathbf{z}}(\mathbf{z}) \mid\mid p_{\mathbf{z}}(\mathbf{z}))$
- Variational Inference : Minimize $\mathrm{KL}(p_{\mathbf{z}}(\mathbf{z}) \mid\mid p^*_{\mathbf{z}}(\mathbf{z}))$

One can show that fitting the model $p_{\mathbf{z}}(\mathbf{z})$ to the target $p^*_{\mathbf{z}}(\mathbf{z})$ via $\mathrm{KL}(p^*_{\mathbf{z}}(\mathbf{z}) \mid\mid p_{\mathbf{z}}(\mathbf{z}))$ (ML) is equivalent to fitting $p^*_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ to the base $p_{\boldsymbol{\epsilon}}(\boldsymbol{\epsilon})$ and vice versa.

Intuition



g
$p_\varepsilon(\varepsilon)$
$p_z(z)$
≈
$p^*_\varepsilon(\varepsilon)$
$p^*_z(z)$
g^{-1}

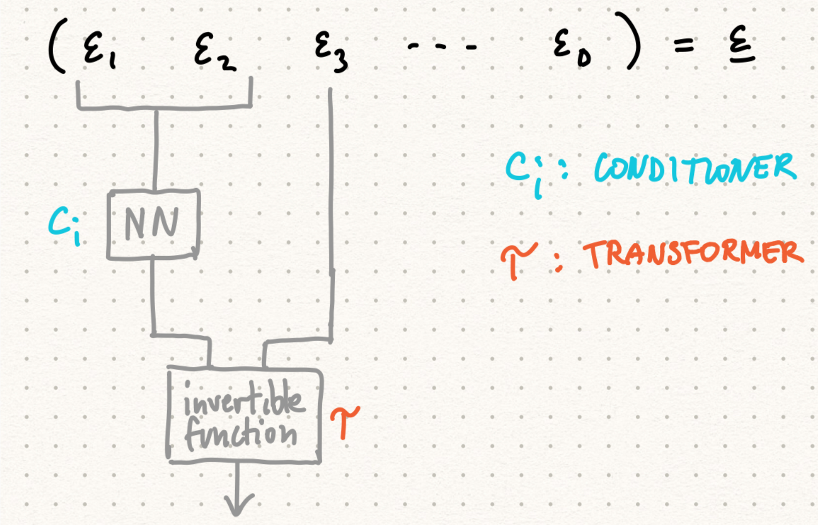
Popular Flows

Triangular Jacobians

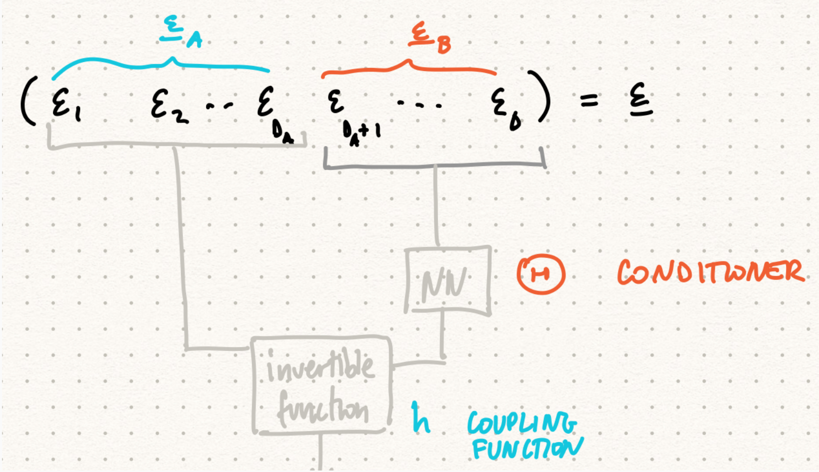
Most popular Normalizing Flows have triangular Jacobians.

- Autoregressive Flows: Feed previous coordinates into an arbitrarily complex neural network. The output is then fed, together with the currect coordinate into an invertible transformation.
- Coupling Flows: Coordinates are partitioned in two $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_A, \boldsymbol{\epsilon}_B)$. The second partition $\boldsymbol{\epsilon}_B$ is fed into an arbitrarily complex function and then, the output is fed into an invertible function together with $\boldsymbol{\epsilon}_A$.

Autoregressive Flows



Coupling Flows



Thank you

References i