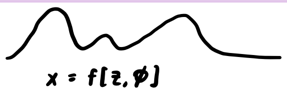
UDL NORMALIZING FLOW NOTES

- 1D ex:

$z \sim \mathcal{N}(0,1)$

simple base distr.

$\xrightarrow{\text{normalizing flow model}}$



desired distr. = $Pr(x)$
↳ (training data distr.)

(where param ϕ are chosen s.t. $Pr(x)$ has the desired distribution)

↳ ★ To generate a new ex. x^*, draw $z^* \sim$ base distr. $\mathcal{N}(0,1)$ & pass thru learned fxn f : $x^* = f[z^*, \phi]$

★ Math behind transforming probability distr.'s :

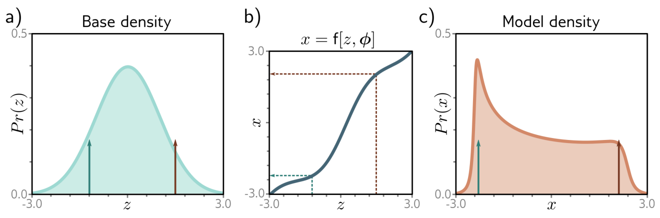


Figure 16.1 Transforming probability distributions. a) The base density is a standard normal defined on a latent variable z. b) This variable is transformed by a function $x = f[z, \phi]$ to a new variable x, which c) has a new distribution. To sample from this model, we draw values z from the base density (green and brown arrows in panel (a) show two examples). We pass these through the function $f[z, \phi]$ as shown by dotted arrows in panel (b) to generate the values of x, which are indicated as arrows in panel (c).

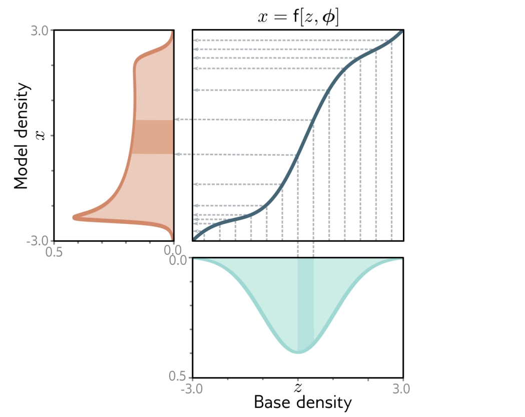


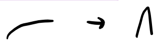
Figure 16.2 Transforming distributions. The base density (cyan, bottom) passes through a function (blue curve, top right) to create the model density (orange, left). Consider dividing the base density into equal intervals (gray vertical lines). The probability mass between adjacent lines must remain the same after transformation. The cyan-shaded region passes through a part of the function where the gradient is larger than one, so this region is stretched. Consequently, the height of the orange-shaded region must be lower so that it retains the same area as the cyan-shaded region. In other places (e.g., $z = -2$), the gradient is less than one, and the model density increases relative to the base density.

> If $\frac{\partial f[z,\phi]}{\partial z} > 1$, small changes in z
$\Downarrow$
large changes in f

$\underbrace{\hphantom{xxxxxxxxxxxx}}$ stretches $Pr(f[z,\phi])$ distr.

> If $\frac{\partial f[z,\phi]}{\partial z} < 1$, large changes in z
$\Downarrow$
small changes in f

$\underbrace{\hphantom{xxxxxxxxxxxx}}$ compresses $Pr(f[z,\phi])$ distr.



Local linear approx. : $f(z + \Delta z) \approx f(z) + \frac{\partial f(z)}{\partial z} \Delta z$

↳ Under $x = f(z)$: $[z, z + \Delta z] \mapsto [f(z), f(z + \Delta z)]$
$\qquad \qquad \qquad \qquad \qquad ||$
$[x, x + \Delta x]$ where $\Delta x \approx f'(z) \Delta z$

★ Probability density : Probability mass in a little interval is conserved : $p_z(z)\Delta z \approx p_x(x)\Delta x$

Substitute $\Delta x \approx |f'(z)|\Delta z$: $p_x(x) \approx \frac{p_z(z)\Delta z}{|f'(z)|\Delta z} = \frac{p_z(z)}{|f'(z)|}$

↳ • If $|f'(z)| < 1$ (compressing) : Δx is smaller
$\Rightarrow$ same probability mass squeezed into less x-space
↳ density increases

• If $|f'(z)| > 1$ (stretching) : same prob. mass spread over more x-space → density decreases

UDL formula : $Pr(x \mid \phi) = \left| \frac{\partial f[z, \phi]}{\partial z} \right|^{-1} \cdot \underline{Pr(z)}$
orig. prob. of latent var. under base density

★ Forward & inverse mappings : f should be invertible

- Sampling requires f : $z \sim p_z(z)$, $x = f[z, \phi]$ ⇐ "forward direction"
- Measuring likelihood (how probable data pt. x is acc. to the model) requires f^{-1} : $Pr(x \mid \phi) = \left| \frac{\partial f[z, \phi]}{\partial z} \right|^{-1} \cdot Pr(z)$
↳ "normalizing direction" ∵ maps complex distr. over $x \mapsto$ normal distr. over z

★ Base density usually chosen to be $\mathcal{N}(0,1)$

★ Learning : objective = Find params ϕ that maximize the likelihood of the training data $\{x_i\}_{i=1}^{I}$

↳ $\hat{\phi} = \underset{\phi}{\text{argmax}} \left[\prod_{i=1}^{I} Pr(x_i \mid \phi) \right] = \underset{\phi}{\text{arg min}} \left[\sum_{i=1}^{I} -\log[Pr(x_i \mid \phi)] \right] = \underset{\phi}{\text{arg min}} \left[\sum_{i=1}^{I} \log\left[\left| \frac{\partial f[z_i, \phi]}{\partial z_i} \right| \right] - \log[Pr(z_i)] \right]$

assuming data iid

using $Pr(x \mid \phi)$ derived earlier

- Normalizing flow: general multi-dim. case

★ Setup: $\vec{z} \in \mathbb{R}^D$ w/ base density $Pr(\vec{z})$ | resulting var. $\vec{x} \in \mathbb{R}^D$ has a new distr.
$\vec{x} = \vec{f}[\vec{z}, \vec{\phi}]$ where $\vec{f}$ is a deep network

↳ To sample new $\vec{x}^*$: 1) $\vec{z}^* \sim Pr(\vec{z})$
2) $\vec{x}^* = \vec{f}[\vec{z}^*, \vec{\phi}]$

★ Likelihood of a sample $\vec{x}$ under this distr. $(Pr(\vec{f}[\vec{z}, \vec{\phi}]))$ $= Pr(\vec{x} | \vec{\phi}) = \left| \frac{\partial \vec{f}[\vec{z}, \vec{\phi}]}{\partial \vec{z}} \right|^{-1} \cdot Pr(\vec{z})$

where $\vec{z} = \vec{f}^{-1}[\vec{x}, \vec{\phi}]$ = latent var. associated w/ $\vec{x}$

$\left| \frac{\partial \vec{f}[\vec{z}, \vec{\phi}]}{\partial \vec{z}} \right|$ = $D \times D$ Jacobian mtx. w/ $\frac{\partial f_i[\vec{z}, \vec{\phi}]}{\partial z_j}$ @ position (i,j) ≈ measures Δvolume @ a point in the multivar. fxn

★ Forward mapping w/ a deep neural network ⇒ series of layers $\vec{f}_k[\cdot, \vec{\phi}_k]$ composed together

$$\vec{x} = \vec{f}[\vec{z}, \vec{\phi}] = \vec{f}_K[\vec{f}_{K-1}[\cdots \vec{f}_2[\vec{f}_1[\vec{z}, \vec{\phi}_1], \vec{\phi}_2], \ldots \vec{\phi}_{K-1}], \vec{\phi}_K]$$

Inverse mapping: $\vec{z} = \vec{f}^{-1}[\vec{x}, \vec{\phi}] = \vec{f}_1^{-1}[\vec{f}_2^{-1}[\cdots \vec{f}_{K-1}^{-1}[\vec{f}_K^{-1}[\vec{x}, \vec{\phi}_K], \vec{\phi}_{K-1}], \cdots \vec{\phi}_2], \vec{\phi}_1]$

★ Called "Normalizing flows" ∵ base density is usually chosen to be a multivariate standard normal $\mathcal{N}(\vec{0}, I)$
↳ Effect of each subsequent inverse layer is to gradually move ("flow") the data density toward this $\mathcal{N}$ distr.

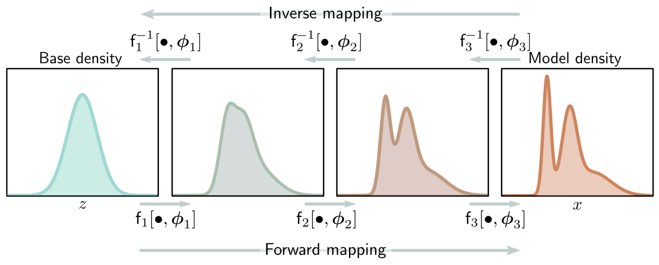


Figure 16.4 Forward and inverse mappings for a deep neural network. The base density (left) is gradually transformed by the network layers $f_1[\bullet, \phi_1], f_2[\bullet, \phi_2], \ldots$ to create the model density. Each layer is invertible, and we can equivalently think of the inverse of the layers as gradually transforming (or "flowing") the model density back to the base density.

★ Chain rule expanding $\frac{\partial \vec{f}[\vec{z}, \vec{\phi}]}{\partial \vec{z}}$: Define $\vec{x}_k = \vec{f}_k[\cdot, \vec{\phi}_k]$ (Recall $\vec{f}[\vec{z}, \vec{\phi}] = \vec{f}_K[\vec{f}_{K-1}[\cdots \vec{f}_2[\vec{f}_1[\vec{z}, \vec{\phi}_1], \vec{\phi}_2], \ldots \vec{\phi}_{K-1}], \vec{\phi}_K]$)

$$\therefore \frac{\partial \vec{f}[\vec{z}, \vec{\phi}]}{\partial \vec{z}} = \frac{\partial \vec{f}_K[\vec{x}_{K-1}, \vec{\phi}_K]}{\partial \vec{x}_{K-1}} \frac{\partial \vec{f}_{K-1}[\vec{x}_{K-2}, \vec{\phi}_{K-1}]}{\partial \vec{x}_{K-2}} \cdot \ldots \cdot \frac{\partial \vec{f}_1[\vec{z}, \vec{\phi}_1]}{\partial \vec{z}} = \frac{\partial \vec{x}_K}{\partial \vec{x}_{K-1}} \frac{\partial \vec{x}_{K-1}}{\partial \vec{x}_{K-2}} \cdot \ldots \cdot \frac{\partial \vec{x}_1}{\partial \vec{z}}$$

↳ Determinant $\left| \frac{\partial \vec{f}[\vec{z}, \vec{\phi}]}{\partial \vec{z}} \right|$ obtained via $\det(A \cdot B) = \det(A)\det(B)$

★ Training: Dataset $\{\vec{x}_i\}$ of I training examples; neg. log-likelihood

$$\hat{\vec{\phi}} = \underset{\vec{\phi}}{\arg\max} \left[\prod_{i=1}^{I} Pr(\vec{z}_i) \cdot \left| \frac{\partial \vec{f}[\vec{z}_i, \vec{\phi}]}{\partial \vec{z}_i} \right|^{-1} \right]$$

$$= \underset{\vec{\phi}}{\arg\min} \left[\sum_{i=1}^{I} \log\left[\left| \frac{\partial \vec{f}[\vec{z}_i, \vec{\phi}]}{\partial \vec{z}_i} \right| \right] - \log[Pr(\vec{z}_i)] \right]$$ where $\vec{z}_i = \vec{f}^{-1}[\vec{x}_i, \vec{\phi}]$

★ Desiderata for network layers $\vec{f}_k$:
1) $\vec{f}_K[\vec{f}_{K-1}[\cdot \cdots \cdot], \vec{\phi}_K]$ should be sufficiently expressive
2) Network layers must be invertible (bijection)
3) Must be possible to compute the inverse of each layer efficiently
↳ $\vec{z}_i = \vec{f}^{-1}[\vec{x}_i, \vec{\phi}]$ occurs repeatedly for likelihood computation during training
4) Must be possible to compute det. of Jacobian efficiently

- Invertible network layers
  - ⋆ Linear: $\vec{f}[\vec{h}] = \vec{\beta} + \Omega\vec{h}$
  - ⋆ Elementwise: apply pointwise nonlin. fxn $f[\cdot, \phi]$ to each element of input: $\vec{f}[\vec{h}] = [f[h_1, \phi], \ldots, f[h_D, \phi]]^T$

    efficient but usually not expressive enough

  - ⋆ Coupling flows
  - ⋆ Autoregressive flows
  - ⋆ Inverse autoregressive flows
  - ⋆ Residual flows (iRevNet, iResNet)

  Read more!

- Multi-scale flows   Read more!

- Applications:
  - ⋆ Only model (Flow) out of GANs, VAEs, diffusion that can compute the exact log-likelihood of a new sample

  - ⋆ Modeling densities
  - ⋆ Synthesis
  - ⋆ Approximating other density models