Normalizing Flows (contd.)

- defined coupling layers: $z^{(\ell)}_{\le \tilde{D}} = z^{(\ell-1)}_{\le \tilde{D}}$, $z^{(\ell)}_{>\tilde{D}} = s_\ell\left(z^{(\ell-1)}_{\le \tilde{D}}\right) \overset{\text{element-wise}}{\cdot} z^{(\ell-1)}_{>\tilde{D}} + t_\ell\left(z^{(\ell-1)}_{\le \tilde{D}}\right)$ $\quad \tilde{D} = \left\lfloor \frac{D}{2} \right\rfloor$
  - fast forward computation:
    needed for training & for inference ($\hat{=}$ compute $\hat{p}(X=x)$)
  - fast backward computation: $z^{(\ell-1)}_{\le \tilde{D}} = z^{(\ell)}_{\le \tilde{D}}$, $z^{(\ell-1)}_{>\tilde{D}} = \left(z^{(\ell)}_{>\tilde{D}} - t_\ell\left(z^{(\ell)}_{\le \tilde{D}}\right)\right) / s_\ell\left(z^{(\ell)}_{\le \tilde{D}}\right)$
    needed for data generation
    important: the nested functions $s_\ell(\cdot)$ and $t_\ell(\cdot)$ are always executed forward
    $\Rightarrow$ constructed an invertible layer, using non-invertible networks $s_\ell(\cdot), t_\ell(\cdot)$
  - cheap Jacobian determinant: $\det\left(f'\left(z^{(\ell-1)}\right)\right) = \prod_{j=\tilde{D}+1}^{D} s_{\ell j}\left(z^{(\ell-1)}_{\le \tilde{D}}\right)$
    needed for training & inference

- To build a deep network from coupling layers, we must ensure that the skip connections are not always the same dimensions ⇒ introduce orthonormal matrices Q_ℓ between layers

$$f(x) = f_L \circ Q_{L-1} \circ f_{L-1} \circ Q_{L-2} \circ \dots \circ Q_1 \circ f_1$$

orthonormal is nice, because: - easily inverted: $Q^{-1} = Q^T$

- Jacobian determinant: $|\det Q| = 1$

two basic choices: - Q = random_shuffle_columns($\mathbb{I}$) → permutation matrix

- variant: for ℓ odd: $Q = \begin{pmatrix} 0 & \mathbb{I}_{\tilde{D}} \\ \mathbb{I}_{D-\tilde{D}} & 0 \end{pmatrix}$ exchanges first & second half of $z^{(\ell-1)}$

⇒ every dimensions has been transformed after two couplings

- Q is a random rotation: A = random_normal(D, D)

QR decomposition: $A = QR$, use Q

$\boxed{\text{learning of } Q \text{ does not improve models in practice}}$

for coupling layers to be invertible, $s_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}}) \neq 0$

without loss of generality, we require: $s_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}}) > 0$

$\Rightarrow$ actually learn $\tilde{s}_{\ell j}(z^{(\ell-1)}_{\leq D})$ unconstrained, $s_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}}) = \exp(\tilde{s}_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}}))$

in practice, for numerical stability, we constrain $\frac{1}{\varepsilon} \leq s < \varepsilon$.

for example $s_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}}) = \exp(a \cdot \tanh(\tilde{s}_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}})))$

$$\Rightarrow \log \det f'_\ell(z^{(\ell-1)}) = \sum_{j=\tilde{D}+1}^{D} \tilde{s}_{\ell j}(z^{(\ell-1)}_{\leq \tilde{D}})$$

training of a NF: we want to minimize forward KL: $KL[p^*(x) \| \hat{p}(x)] = 0$

$$KL[p^*(x) \| p(x)] = \int p^*(x) \log \frac{p^*(x)}{p(x)} dx = \int p^*(x) (-\log p(x))\, dx - \underbrace{\int p^*(x) (-\log p^*(x))\, dx}_{H(p^*) \text{ entropy of data, independent of model} \Rightarrow \text{drop}}$$

minimal NLL loss: $= \mathbb{E}_{x \sim p^*(x)} [-\log p(x)] + \text{const}$

$$\approx \frac{1}{N} \sum_{i=1}^{N} -\log p(x_i) \overset{\text{change of variables}}{=} \frac{1}{N} \sum_{i=1}^{N} -\log \underbrace{q(f(x_i))}_{\text{desired code distribution} = \mathcal{N}(0, \mathbb{I})} - \log \det f'(x_i)$$

if $q(z) \sim \mathcal{N}(0, \mathbb{I}) \Rightarrow -\log q(f(x_i)) = \underbrace{-\log \text{normalization}}_{\text{const.} \Rightarrow \text{drop}} + \frac{f(x_i)^2}{2}$

if $f(x_i)$ is a coupling network: $\log \det f'(x_i) = \sum_{l=1}^{L} \sum_{j=\tilde{D}+1}^{D} \tilde{s}_{lj}(z^{(l-1)}_{\leq \tilde{D}})$

training objective of NF with $q(z) = \mathcal{N}(0, \mathbb{I})$:

$$\boxed{\hat{f}(x) = \arg\min_{f} \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\|f(x_i)\|^2}{2} - \sum_{l=1}^{L} \sum_{j=\tilde{D}+1}^{D} \tilde{s}_{lj}\left(\tilde{z}^{(l-1)}_{i, \le \tilde{D}} \right) \right)}$$

alg: (0) initialize s_l, t_l randomly as in ordinary neural networks, init Q_l randomly (not trainable)

(1) for $t = 1, \dots, T$

do a gradient step of loss to update s_l, t_l (ordinary back-prop, use ADAM)

known as "Real NVP" network (NVP $\hat{=}$ non-volume-preserving $\hat{=}$ $\det f'(x) \neq 1$)

[Dinh et al. 2017]

universal, e.g. [Draxler et al. 2022–20

implemented in various libraries, e.g. FrEIA, BayesFlow [.org]

other

- simplification: volume-preserving NFs: $\det f'(x) = 1$ for all x

  - NICE architecture [Dinh et al. 2015]

$$z^{(l)}_{>\tilde{D}} = z^{(l-1)}_{>\tilde{D}} + t_l\left(z^{(l-1)}_{\le \tilde{D}}\right)$$

$$\Rightarrow f_l'\left(z^{(l-1)}\right) = \mathbb{I}, \quad \det f_l'\left(z^{(l-1)}\right) = 1$$

  - GIN architecture [Sorrenson et al. 2020] "general incompressible NF"

$$z^{(l)}_{>\tilde{D}} = \exp\left(\tilde{\tilde{s}}_l\left(z^{(l-1)}_{\le \tilde{D}}\right)\right) \cdot z^{(l-1)}_{>\tilde{D}} + t_l\left(z^{(l-1)}_{\le \tilde{D}}\right)$$

$$\tilde{\tilde{s}}_l\left(z^{(l-1)}_{\le \tilde{D}}\right) = \tilde{s}_l\left(z^{(l-1)}_{\le \tilde{D}}\right) - \operatorname{mean}_{j=\tilde{D}+1 \ldots D} \tilde{s}_l\left(z^{(l-1)}_{\le \tilde{D}}\right)$$

$$\log \det f_l'\left(z^{(l-1)}\right) = \sum_{j=\tilde{D}+1}^{D} \tilde{\tilde{s}}_{l,j}\left(z^{(l-1)}_{\le \tilde{D}}\right) = 0 \Rightarrow \det f'(\,) = 1$$

- advantage: $\hat{p}(X = x) = q(Z = f(x))$, e.g. simplifies power scaling $\hat{p}(x)^{\beta}$

disadvantage: volume-preserving NFs are not universal, e.g. cannot change # modes

- if X is an image: GLOW architecture [Kingma & Dhariwal 2018]
  - $s_\ell(\cdot)$ and $t_\ell(\cdot)$ are convolutional networks
  - active and passive part of $z^{(\ell+1)}$ is selected among the channels (not locations)
- Wavelet Flow: decompose the image hierarchically using Wavelets (similar to Fourier transform)

next time: how to learn conditional distributions $p(X \mid Y)$

hidden properties → X; Y ← observations