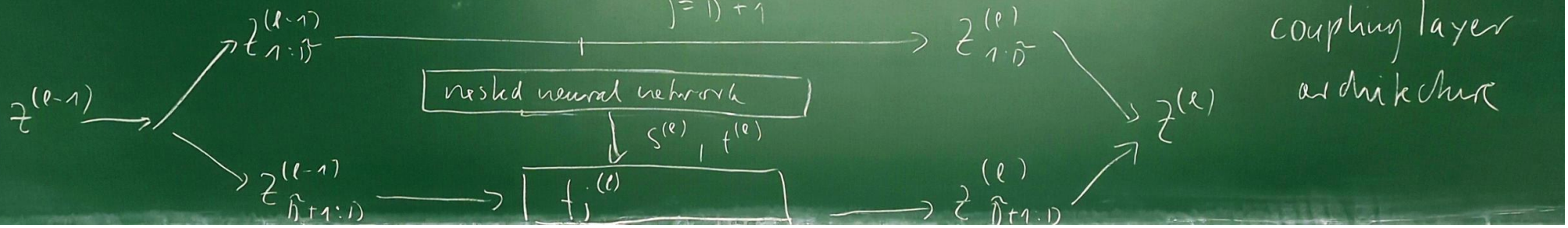
Normalizing Flows (contd.)

- recap: Real NVP, invertible neural network (INN)
  - coupling layers with affine (= easily invertible) transformations, with autoregressive form ⇒ Jacobian determinant easy

layer ℓ:

$$z^{(\ell)} = f^{(\ell)}(z^{(\ell-1)}) \begin{cases} z_j^{(\ell)} = z_j^{(\ell-1)} & \text{for } j = 1, \dots, \tilde{D} \qquad \tilde{D} = \left\lfloor \frac{D}{2} \right\rfloor \\ z_\ell^{(\ell)} = \exp\left(s_j^{(\ell)}(z_{1:\tilde{D}}^{(\ell-1)})\right) \cdot z_j^{(\ell-1)} + t_j^{(\ell)}(z_{1:\tilde{D}}^{(\ell-1)}) & \text{for } j = \tilde{D}+1, \dots, D \end{cases}$$

$$\det\left(\frac{\partial f^{(\ell)}(z^{(\ell-1)})}{\partial z^{(\ell-1)}}\right) = \prod_{j=\tilde{D}+1}^{D} \exp\left(s_j^{(\ell)}(z_{1:\tilde{D}}^{(\ell-1)})\right)$$



coupling layer architecture

- turn this into a deep network, many coupling layers

  naive: $z_{1:\tilde{D}}$ are always skip connections $p_E(z_{1:\tilde{D}}) = p^*(x_{1:\tilde{D}}) \neq q(z_{1:\tilde{D}})$

⇒ idea: change dimensions in skip part

after each coupling, add an ① index permutation (special case of ②)

② orthonormal transformation Q

orthonormal is good because $|\det(Q)| = 1$, easily invertible: $Q^{-1} = Q^T$

- experimentally, it turned out that learning Q is not necessary, random fixed matrices are sufficient (may change with recent new learning alg. for orthonormal matrices)

how to create Q?
- create a random Gaussian matrix A, $A_{ij} = N(0,1)$
- $Q, R = qr\text{decomposition}(A)$

final architecture $f = f^{(L)} \circ Q^{(L-1)} \circ f^{(L-1)} \circ \dots \circ Q^{(1)} \circ f^{(1)}$

$f^{(l)}$ coupling layers

$Q^{(l)}$ random orthonorm.

- training algorithm: minimize negative log-likelihood of data

$$\hat{f} = \arg\min_f \mathbb{E}_{x \sim p^*(x)} \left[-\log p_f(x) \right] \qquad p_f(x) = q(z = f(x)) \det \left(J_f \right)$$

$$\approx \arg\min_f \frac{1}{N} \sum_{i=1}^{N} -\log p_f(x_i) = \arg\min_f \frac{1}{N} \sum_{i=1}^{N} -\log q(f(x_i)) - \log \det \left(J_f(x_i) \right)$$

if $q(z) = \mathcal{N}(0, \mathbb{I})$ $\Rightarrow$ $-\log q(z) = \frac{z^2}{2} + \log (2\pi)^{D/2}$

if $\det |J_f|$ is $\prod_{l=1}^{L} \prod_{j=\tilde{D}+1}^{D} \underbrace{s_j^{(l)}\left(z_{1:\tilde{D}}^{(l-1)}\right)}_{\exp(\tilde{s})}$ $\Rightarrow$ $\log \det \left(J_f \right) = \sum_{l=1}^{L} \sum_{j=\tilde{D}+1}^{D} \tilde{s}_j^{(l)}\left(z_{1:\tilde{D}}^{(l-1)}\right)$

$$\hat{f} = \arg\min_f \frac{1}{N} \sum_{i=1}^{N} \left(\frac{f(x_i)^2}{2} - \sum_{l=1}^{L} \sum_{j=\tilde{D}+1}^{D} \tilde{s}_j^{(l)}\left(z_{i,1:\tilde{D}}^{(l-1)}\right) \right)$$

• why does affine coupling work?

$p(z^{(\ell-1)}) = N(0, \Sigma)$ $\Rightarrow$ want that $p(z^{(\ell)})$ is closer to $N(0, \mathbb{I})$ than $p(z^{(\ell-1)})$

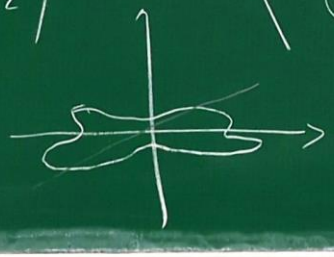
$$KL\left[p(z^{(\ell)}) \,\|\, N(0,\mathbb{I})\right] \approx \tfrac{1}{2} KL\left[p(z^{(\ell-1)}) \,\|\, N(0,\mathbb{I})\right]$$

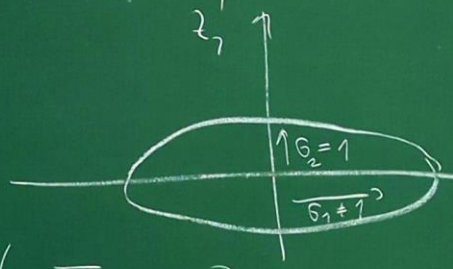
holds empirically for most $p(z^{(\ell-1)})$, proven for $p(z^{(\ell-1)}) = N(0, \Sigma)$

before:

$\leftarrow$ coupling scales & translates every slice to mean = 0 variance = 1

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \rightsquigarrow \begin{pmatrix} \Sigma_{11} & 0 \\ 0 & \Sigma_{22} - (\Sigma_{12}\Sigma_{11}^{-1}\Sigma_{21}) \end{pmatrix}$$



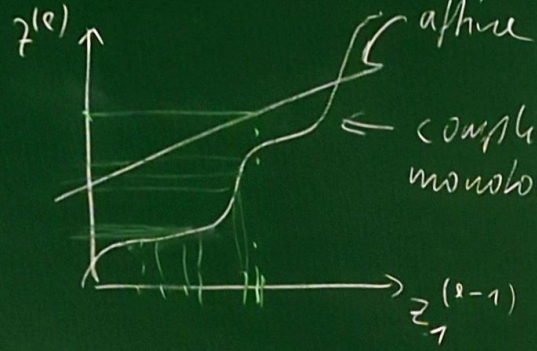


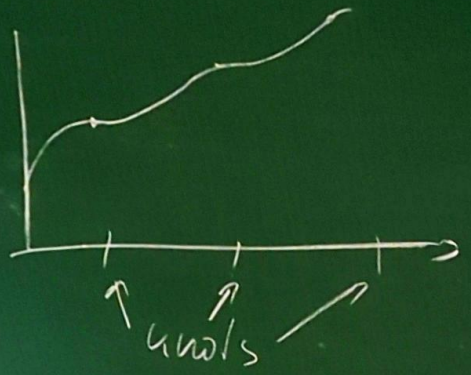
random rotate and repeat

after many random rotations enough "good slices" are encountered to get a standard normal

- spline couplings are superior to affine couplings in low dimensions ($D < 20$)

$$f_j^{(i)} = s_j^{(i)} \, z_j + t_j^{(i)}$$





Hermite splines:
- location of knots
- function at knots
- derivatives at knots

$\Rightarrow$ spline interpolates in between

$z^{(i-1)}$ ↳ $z^{(i)}$ ↳ $f'^{(i)}$ ↳

→ nested networks learn these parameters & derive some spline to interpolate

ideally

$$\Phi^{-1}\left(\mathrm{CDF}\left(z_j^{(i-1)} \mid z_{1:D}^{(i-1)}\right)\right)$$

↑ inverse CDF of standard normal

affine approximates as straight line
spline as a piecewise simple fct.

most popular: rational quadratic spline

"Neural spline Flows" [2019] $\frac{a z^2 + b z + c}{d z^2 + e \cdot z + g}$

- various libraries (e.g. FrEIA in pytorch, Bayesflow in tensorflow) implement INNs
- major variant for images: nested networks are convolutional "Glow" [Kingma & Dhariwal 2018]
  funny face manipulation demo in the web
  new: actnorm layers ≙ adapt batchnorm to normalizing flows
- better variants of normalizing flows
  - diffusion models (e.g. "stable diffusion"): implement $f(x)$ by a differential equation
  - free-form flows [Draxler et al. 2023]: an to encoder without bottleneck
    ($\dim(z) = \dim(x)$) ⇒ can be any network, no couplings or other restrictions

) later