

# Deep Generative Models

## Lecture 5

Roman Isachenko



AI Masters

Autumn, 2022

# Recap of previous lecture

## LVM

$$p(\mathbf{x}|\theta) = \int p(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}, \theta) p(\mathbf{z}) d\mathbf{z}$$

- ▶ More powerful  $p(\mathbf{x}|\mathbf{z}, \theta)$  leads to more powerful generative model  $p(\mathbf{x}|\theta)$ .
- ▶ Too powerful  $p(\mathbf{x}|\mathbf{z}, \theta)$  could lead to posterior collapse:  $q(\mathbf{z}|\mathbf{x}, \phi)$  will not carry any information about  $\mathbf{x}$  and close to prior  $p(\mathbf{z})$ .

## Autoregressive decoder

$$p(\mathbf{x}|\mathbf{z}, \theta) = \prod_{j=1}^m p(x_j | \mathbf{x}_{1:j-1}, \mathbf{z}, \theta)$$

- ▶ Global structure is captured by latent variables  $\mathbf{z}$ .
- ▶ Local statistics are captured by limited receptive field of

autoregressive context  $\mathbf{x}_{1:j-1}$ .

# Recap of previous lecture

## Decoder weakening

- ▶ Powerful decoder  $p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta})$  makes the model expressive, but posterior collapse is possible.
- ▶ PixelVAE model uses the autoregressive PixelCNN model with small number of layers to limit receptive field.

## KL annealing

$$\mathcal{L}(\phi, \boldsymbol{\theta}, \beta) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \beta \cdot KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z}))$$

Start training with  $\beta = 0$ , increase it until  $\beta = 1$  during training.

## Free bits

Ensure the use of less than  $\lambda$  bits of information:

$$\mathcal{L}(\phi, \boldsymbol{\theta}, \lambda) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) - \max(\lambda, KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z}))).$$

This results in  $KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z})) \geq \lambda$ .

# Recap of previous lecture

## VAE objective

$$\log p(\mathbf{x}|\boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta}) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x}, \phi)} \log \frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \phi)} \rightarrow \max_{q, \boldsymbol{\theta}}$$

## IWAE objective

$$\mathcal{L}_K(q, \boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}_1, \dots, \mathbf{z}_K \sim q(\mathbf{z}|\mathbf{x}, \phi)} \log \left( \frac{1}{K} \sum_{k=1}^K \frac{p(\mathbf{x}, \mathbf{z}_k|\boldsymbol{\theta})}{q(\mathbf{z}_k|\mathbf{x}, \phi)} \right) \rightarrow \max_{\phi, \boldsymbol{\theta}}.$$

## Theorem

1.  $\log p(\mathbf{x}|\boldsymbol{\theta}) \geq \mathcal{L}_K(q, \boldsymbol{\theta}) \geq \mathcal{L}_M(q, \boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta})$ , for  $K \geq M$ ;
  2.  $\log p(\mathbf{x}|\boldsymbol{\theta}) = \lim_{K \rightarrow \infty} \mathcal{L}_K(q, \boldsymbol{\theta})$  if  $\frac{p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}, \phi)}$  is bounded.
- ▶ IWAE makes the variational bound tighter and extends the class of variational distributions.
  - ▶ Gradient signal becomes really small, training is complicated.
  - ▶ IWAE is a standard quality measure for VAE models.

# Recap of previous lecture

## Jacobian matrix

Let  $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a differentiable function.

$$\mathbf{z} = f(\mathbf{x}), \quad \mathbf{J} = \frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \cdots & \frac{\partial z_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial z_m}{\partial x_1} & \cdots & \frac{\partial z_m}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{m \times m}$$

## Change of variable theorem (CoV)

Let  $\mathbf{x}$  be a random variable with density function  $p(\mathbf{x})$  and  $f : \mathbb{R}^m \rightarrow \mathbb{R}^m$  is a differentiable, invertible function (diffeomorphism). If  $\mathbf{z} = f(\mathbf{x})$ ,  $\mathbf{x} = f^{-1}(\mathbf{z}) = g(\mathbf{z})$ , then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J}_f)| = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x})) \left| \det \left( \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$
$$p(\mathbf{z}) = p(\mathbf{x}) |\det(\mathbf{J}_g)| = p(\mathbf{x}) \left| \det \left( \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(g(\mathbf{z})) \left| \det \left( \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}} \right) \right|.$$

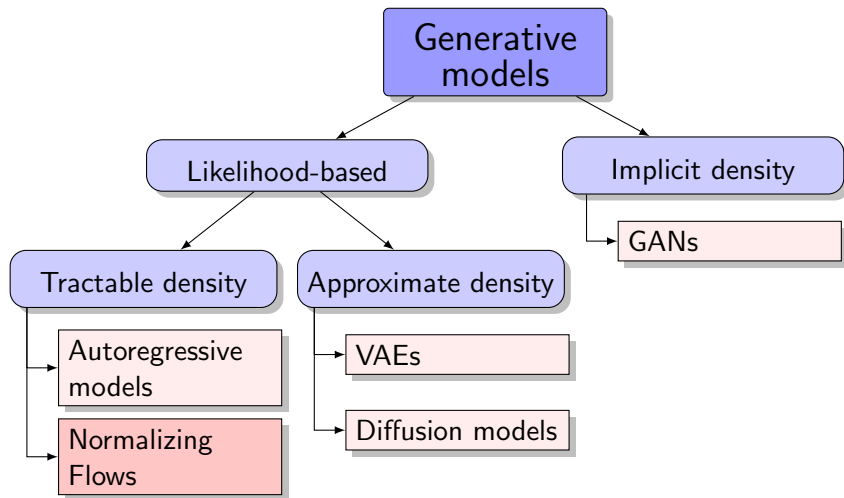
# Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Linear flows
4. Autoregressive flows
5. Inverse autoregressive flows

# Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Linear flows
4. Autoregressive flows
5. Inverse autoregressive flows

# Generative models zoo





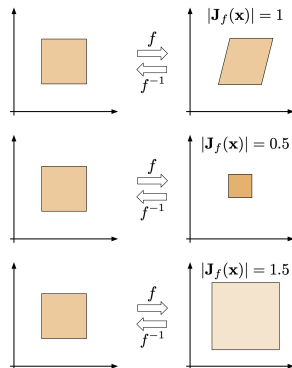
# Jacobian determinant

## Inverse function theorem

If function  $f$  is invertible and Jacobian matrix is continuous and non-singular, then

$$\mathbf{J}_f = \mathbf{J}_{g^{-1}} = \mathbf{J}_g^{-1}; \quad |\det(\mathbf{J}_f)| = \frac{1}{|\det(\mathbf{J}_g)|}.$$

- ▶  $\mathbf{x}$  and  $\mathbf{z}$  have the same dimensionality ( $\mathbb{R}^m$ ).
- ▶  $f(\mathbf{x}, \boldsymbol{\theta})$  could be parametric function.
- ▶ Determinant of Jacobian matrix  $\mathbf{J} = \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}}$  shows how the volume changes under the transformation.

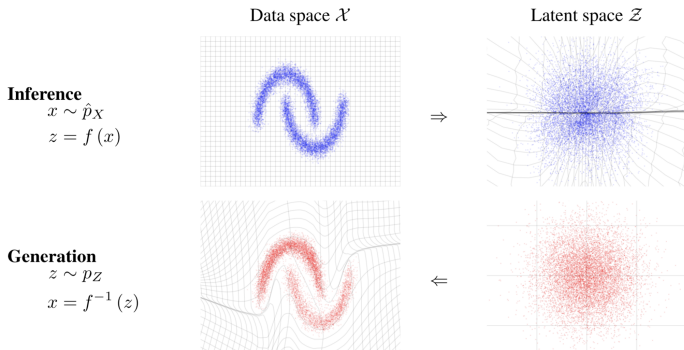


# Fitting flows

## MLE problem

$$p(\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x}, \boldsymbol{\theta})) \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)| \rightarrow \max_{\boldsymbol{\theta}}$$

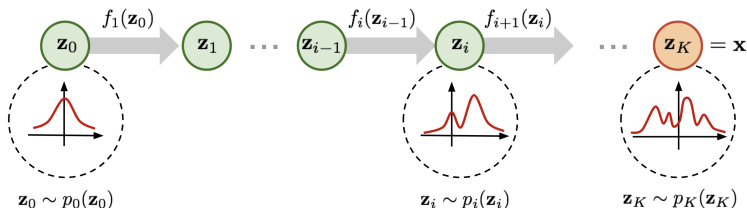


# Composition of flows

## Theorem

Diffeomorphisms are **composable** (If  $\{f_k\}_{k=1}^K$  satisfy conditions of the change of variable theorem, then  $\mathbf{z} = f(\mathbf{x}) = f_K \circ \dots \circ f_1(\mathbf{x})$  also satisfies it).

$$\begin{aligned} p(\mathbf{x}) &= p(f(\mathbf{x})) \left| \det \left( \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = p(f(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}_K}{\partial \mathbf{f}_{K-1}} \dots \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}} \right) \right| = \\ &= p(f(\mathbf{x})) \prod_{k=1}^K \left| \det \left( \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right| = p(f(\mathbf{x})) \prod_{k=1}^K |\det(\mathbf{J}_{f_k})| \end{aligned}$$



# Flows

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

## Definition

Normalizing flow is a *differentiable, invertible* mapping from data  $\mathbf{x}$  to the noise  $\mathbf{z}$ .

- ▶ **Normalizing** means that the inverse flow takes samples from  $\pi(\mathbf{x})$  and normalizes them into samples from the density  $p(\mathbf{z})$ .
- ▶ **Flow** refers to the trajectory followed by samples from  $p(\mathbf{z})$  as they are transformed by the sequence of transformations

$$\mathbf{z} = f_K \circ \dots \circ f_1(\mathbf{x}); \quad \mathbf{x} = f_1^{-1} \circ \dots \circ f_K^{-1}(\mathbf{z}) = g_1 \circ \dots \circ g_K(\mathbf{z})$$

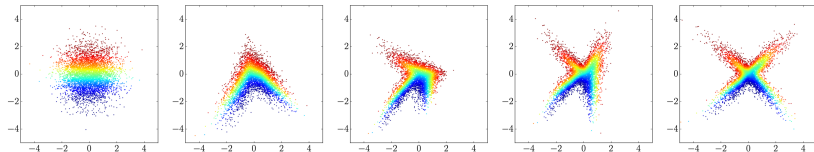
## Log likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f_K \circ \dots \circ f_1(\mathbf{x})) + \sum_{k=1}^K \log |\det(\mathbf{J}_{f_k})|,$$

where  $\mathbf{J}_{f_k} = \frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}}$ .

# Flows

## Example of a 4-step flow



## Flow log likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

What is the complexity of the determinant computation?

What we need:

- ▶ efficient computation of the Jacobian matrix  $\mathbf{J}_f = \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}}$ ;
- ▶ efficient inversion of  $f(\mathbf{x}, \boldsymbol{\theta})$ ;
- ▶ loss function to minimize.

# Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Linear flows
4. Autoregressive flows
5. Inverse autoregressive flows

# Forward KL vs Reverse KL

## Forward KL

$$\begin{aligned} KL(\pi||p) &= \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x}|\boldsymbol{\theta})} d\mathbf{x} \\ &= -\mathbb{E}_{\pi(\mathbf{x})} \log p(\mathbf{x}|\boldsymbol{\theta}) + \text{const} \rightarrow \min_{\boldsymbol{\theta}} \end{aligned}$$

Maximum likelihood estimation is equivalent to minimization of the Monte-Carlo estimation of forward KL.

## Forward KL for flow model

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log |\det(\mathbf{J}_f)|$$

- ▶ We need to be able to compute  $f(\mathbf{x}, \boldsymbol{\theta})$  and its Jacobian.
- ▶ We need to be able to compute the density  $p(\mathbf{z})$ .
- ▶ We don't need to think about computing the function  $g(\mathbf{z}, \boldsymbol{\theta}) = f^{-1}(\mathbf{z}, \boldsymbol{\theta})$  until we want to sample from the flow.

# Forward KL vs Reverse KL

## Reverse KL

$$\begin{aligned} KL(p||\pi) &= \int p(\mathbf{x}|\boldsymbol{\theta}) \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x})} d\mathbf{x} \\ &= \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})} [\log p(\mathbf{x}|\boldsymbol{\theta}) - \log \pi(\mathbf{x})] \rightarrow \min_{\boldsymbol{\theta}} \end{aligned}$$

## Reverse KL for flow model (LOTUS trick)

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{z}) + \log |\det(\mathbf{J}_f)| = \log p(\mathbf{z}) - \log |\det(\mathbf{J}_g)|$$

$$KL(p||\pi) = \mathbb{E}_{p(\mathbf{z})} [\log p(\mathbf{z}) - \log |\det(\mathbf{J}_g)| - \log \pi(g(\mathbf{z}, \boldsymbol{\theta}))]$$

- ▶ We need to be able to compute  $g(\mathbf{z}, \boldsymbol{\theta})$  and its Jacobian.
- ▶ We need to be able to sample from the density  $p(\mathbf{z})$  (do not need to evaluate it) and to evaluate(!)  $\pi(\mathbf{x})$ .
- ▶ We don't need to think about computing the function  $f(\mathbf{x}, \boldsymbol{\theta})$ .

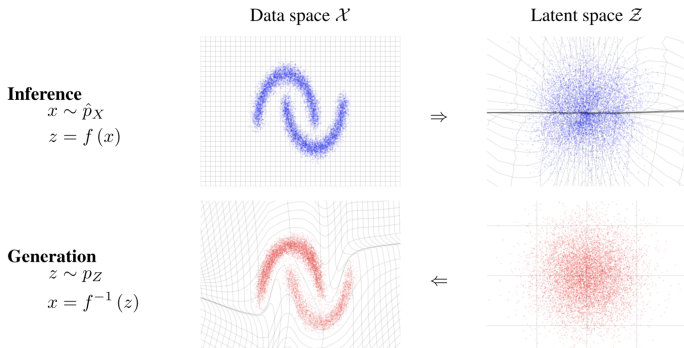


# Flow KL duality

## Theorem

Fitting flow model  $p(\mathbf{x}|\boldsymbol{\theta})$  to the target distribution  $\pi(\mathbf{x})$  using forward KL (MLE) is equivalent to fitting the induced distribution  $p(\mathbf{z}|\boldsymbol{\theta})$  to the base  $p(\mathbf{z})$  using reverse KL:

$$\arg \min_{\boldsymbol{\theta}} KL(\pi(\mathbf{x})||p(\mathbf{x}|\boldsymbol{\theta})) = \arg \min_{\boldsymbol{\theta}} KL(p(\mathbf{z}|\boldsymbol{\theta})||p(\mathbf{z})).$$



# Flow KL duality

## Theorem

$$\arg \min_{\theta} KL(\pi(\mathbf{x})||p(\mathbf{x}|\theta)) = \arg \min_{\theta} KL(p(\mathbf{z}|\theta)||p(\mathbf{z})).$$

## Proof

$$\blacktriangleright \mathbf{z} \sim p(\mathbf{z}), \mathbf{x} = g(\mathbf{z}, \theta), \mathbf{x} \sim p(\mathbf{x}|\theta);$$

$$\blacktriangleright \mathbf{x} \sim \pi(\mathbf{x}), \mathbf{z} = f(\mathbf{x}, \theta), \mathbf{z} \sim p(\mathbf{z}|\theta);$$

$$\log p(\mathbf{z}|\theta) = \log \pi(g(\mathbf{z}, \theta)) + \log |\det(\mathbf{J}_g)|;$$

$$\log p(\mathbf{x}|\theta) = \log p(f(\mathbf{x}, \theta)) + \log |\det(\mathbf{J}_f)|.$$

$$\begin{aligned} KL(p(\mathbf{z}|\theta)||p(\mathbf{z})) &= \mathbb{E}_{p(\mathbf{z}|\theta)} [\log p(\mathbf{z}|\theta) - \log p(\mathbf{z})] = \\ &= \mathbb{E}_{p(\mathbf{z}|\theta)} [\log \pi(g(\mathbf{z}, \theta)) + \log |\det(\mathbf{J}_g)| - \log p(\mathbf{z})] = \\ &= \mathbb{E}_{\pi(\mathbf{x})} [\log \pi(\mathbf{x}) - \log |\det(\mathbf{J}_f)| - \log p(f(\mathbf{x}, \theta))] = \\ &= \mathbb{E}_{\pi(\mathbf{x})} [\log \pi(\mathbf{x}) - \log p(\mathbf{x}|\theta)] = KL(\pi(\mathbf{x})||p(\mathbf{x}|\theta)). \end{aligned}$$

# Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Linear flows
4. Autoregressive flows
5. Inverse autoregressive flows

# Jacobian structure

## Flow log-likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

The main challenge is a determinant of the Jacobian matrix.

What is the  $\det(\mathbf{J})$  in the following cases?

1. Consider a linear layer  $\mathbf{z} = \mathbf{W}\mathbf{x}$ ,  $\mathbf{W} \in \mathbb{R}^{m \times m}$ .
2. Let  $\mathbf{z}$  be a permutation of  $\mathbf{x}$ .
3. Let  $z_j$  depend only on  $x_j$ .

$$\log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right| = \log \left| \prod_{j=1}^m \frac{\partial f'_j(x_j, \boldsymbol{\theta})}{\partial x_j} \right| = \sum_{j=1}^m \log \left| \frac{\partial f'_j(x_j, \boldsymbol{\theta})}{\partial x_j} \right|.$$

4. Let  $z_j$  depend only on  $\mathbf{x}_{1:j}$  (autoregressive dependency).

# Linear flows

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \boldsymbol{\theta} = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}$$

In general, we need  $O(m^3)$  to invert matrix.

## Invertibility

- ▶ Diagonal matrix  $O(m)$ .
- ▶ Triangular matrix  $O(m^2)$ .
- ▶ It is impossible to parametrize all invertible matrices.

## Invertible 1x1 conv

$\mathbf{W} \in \mathbb{R}^{c \times c}$  - kernel of 1x1 convolution with  $c$  input and  $c$  output channels. The computational complexity of computing or differentiating  $\det(\mathbf{W})$  is  $O(c^3)$ . Cost to compute  $\det(\mathbf{W})$  is  $O(c^3)$ . It should be invertible.

# Linear flows

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}, \quad \boldsymbol{\theta} = \mathbf{W}, \quad \mathbf{J}_f = \mathbf{W}$$

## Matrix decompositions

- ▶ LU-decomposition

$$\mathbf{W} = \mathbf{P}\mathbf{L}\mathbf{U},$$

where  $\mathbf{P}$  is a permutation matrix,  $\mathbf{L}$  is lower triangular with positive diagonal,  $\mathbf{U}$  is upper triangular with positive diagonal.

- ▶ QR-decomposition

$$\mathbf{W} = \mathbf{Q}\mathbf{R},$$

where  $\mathbf{Q}$  is an orthogonal matrix,  $\mathbf{R}$  is an upper triangular matrix with positive diagonal.

---

Kingma D. P., Dhariwal P. *Glow: Generative Flow with Invertible 1x1 Convolutions*, 2018

Hoogeboom E., Van Den Berg R., and Welling M. *Emerging convolutions for generative normalizing flows*, 2019

# Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Linear flows
4. Autoregressive flows
5. Inverse autoregressive flows

# Gaussian autoregressive model

Consider an autoregressive model

$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{j=1}^m p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}), \quad p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}) = \mathcal{N}(\mu_j(\mathbf{x}_{1:j-1}), \sigma_j^2(\mathbf{x}_{1:j-1})).$$

Sampling: reparametrization trick

$$x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}), \quad z_j \sim \mathcal{N}(0, 1).$$

Inverse transform

$$z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})}.$$

We have got an invertible and differentiable transform (it is an autoregressive flow with base distribution  $\mathbf{z} = \mathcal{N}(0, 1)^!$ ).



# Gaussian autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})}.$$

Generation function  $g(\mathbf{z}, \boldsymbol{\theta})$  is **sequential**. Inference function  $f(\mathbf{x}, \boldsymbol{\theta})$  is **not sequential**.

## Forward KL for flow model

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(f(\mathbf{x}, \boldsymbol{\theta})) + \log \left| \det \left( \frac{\partial f(\mathbf{x}, \boldsymbol{\theta})}{\partial \mathbf{x}} \right) \right|$$

- ▶ We need to be able to compute  $f(\mathbf{x}, \boldsymbol{\theta})$  and its Jacobian.
- ▶ We need to be able to compute the density  $p(\mathbf{z})$ .
- ▶ We don't need to think about computing the function  $g(\mathbf{z}, \boldsymbol{\theta}) = f^{-1}(\mathbf{z}, \boldsymbol{\theta})$  until we want to sample from the flow.

# Outline

1. Normalizing flows
2. Forward and Reverse KL for Normalizing flows
3. Linear flows
4. Autoregressive flows
5. Inverse autoregressive flows

# Inverse autoregressive flow (IAF)

Let use the following reparametrization:  $\tilde{\sigma} = \frac{1}{\sigma}$ ;  $\tilde{\mu} = -\frac{\mu}{\sigma}$ .

## Gaussian autoregressive flow

$$x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}) = (z_j - \tilde{\mu}_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{x}_{1:j-1})}$$
$$z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})} = \tilde{\sigma}_j(\mathbf{x}_{1:j-1}) \cdot x_j + \tilde{\mu}_j(\mathbf{x}_{1:j-1}).$$

Let just swap  $\mathbf{z}$  and  $\mathbf{x}$ .

## Inverse autoregressive flow

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot z_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1})$$
$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_j = (x_j - \tilde{\mu}_j(\mathbf{z}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{z}_{1:j-1})}.$$

# Inverse autoregressive flow (IAF)

## Gaussian autoregressive flow: $f(\mathbf{x}, \theta)$

$$x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}).$$

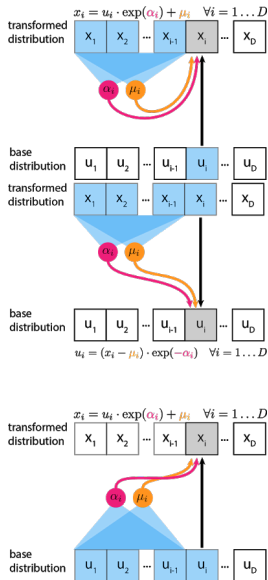
## Inverse transform: $g(\mathbf{z}, \theta)$

$$z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})};$$

$$z_j = \tilde{\sigma}_j(\mathbf{x}_{1:j-1}) \cdot x_j + \tilde{\mu}_j(\mathbf{x}_{1:j-1}).$$

## Inverse autoregressive flow: $f(\mathbf{x}, \theta)$

$$x_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot z_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1}).$$



# Autoregressive flows

## Forward and inverse transforms in MAF

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_j = \sigma_j(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_j(\mathbf{x}_{1:j-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})}.$$

- ▶ Sampling is sequential.
- ▶ Density estimation is parallel.

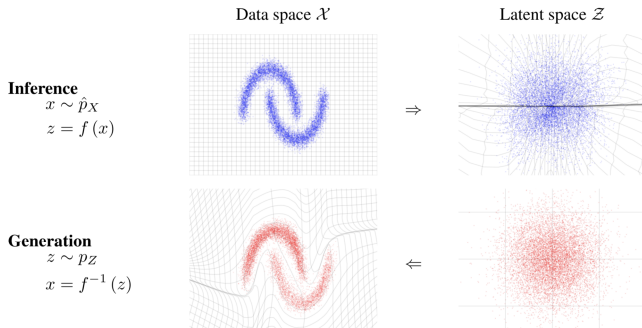
## Forward and inverse transforms in IAF

$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot z_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1}).$$

$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_j = (x_j - \tilde{\mu}_j(\mathbf{z}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{z}_{1:j-1})}.$$

- ▶ Sampling is parallel.
- ▶ Density estimation is sequential.

# Autoregressive flows



- ▶ AF performs parallel inference that is useful for density estimation tasks (forward KL or MLE).
- ▶ IAF performs parallel generation that is useful for optimization of reverse KL.

# Summary

- ▶ Flow models transform a simple base distribution to a complex one via a sequence of invertible transformations with tractable Jacobian.
- ▶ Flow models have a tractable likelihood that is given by the change of variable theorem.
- ▶ Flows could be fitted using forward and reverse KL minimization.
- ▶ Linear flows try to parametrize set of invertible matrices via matrix decompositions.
- ▶ Gaussian autoregressive flow is an autoregressive model with triangular Jacobian.
- ▶ Inverse autoregressive flow is able to sample fast, but the inference is slow.