

# Deep Generative Models

## Lecture 3

Roman Isachenko

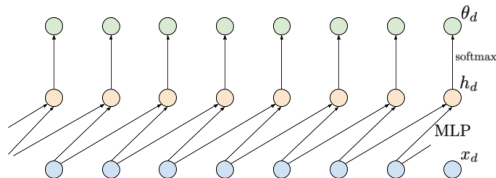


AI Masters

2024, Summer

# Recap of previous lecture

## Autoregressive MLP



## Autoregressive CNN

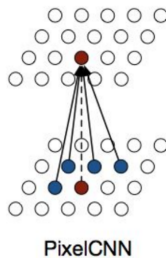
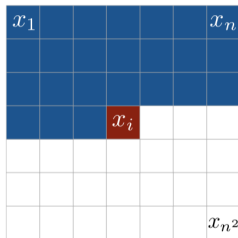


image credit: [https://jmtomczak.github.io/blog/2/2\\_ARM.html](https://jmtomczak.github.io/blog/2/2_ARM.html)

# Recap of previous lecture

## Jacobian matrix

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

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

## Change of variable theorem (CoV)

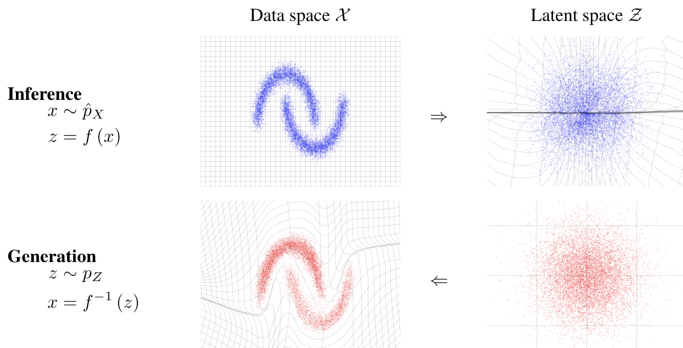
Let  $\mathbf{x}$  be a random variable with density function  $p(\mathbf{x})$  and  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  is a differentiable, invertible function. If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ ,  $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}) = \mathbf{g}(\mathbf{z})$ , then

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

# Recap of previous lecture

## Definition

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



## Log likelihood

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^K \log |\det(\mathbf{J}_{\mathbf{f}_k})|$$

# Recap of previous lecture

## Forward KL for flow model

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

## Reverse KL for flow model

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

## Flow KL duality

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

- ▶  $p(\mathbf{z})$  is a base distribution;  $\pi(\mathbf{x})$  is a data distribution;
- ▶  $\mathbf{z} \sim p(\mathbf{z})$ ,  $\mathbf{x} = \mathbf{g}_\theta(\mathbf{z})$ ,  $\mathbf{x} \sim p(\mathbf{x}|\theta)$ ;
- ▶  $\mathbf{x} \sim \pi(\mathbf{x})$ ,  $\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$ ,  $\mathbf{z} \sim p(\mathbf{z}|\theta)$ .

# Outline

## 1. NF examples

- Linear normalizing flows

- Gaussian autoregressive NF

- RealNVP: coupling layer

# Outline

## 1. NF examples

- Linear normalizing flows

- Gaussian autoregressive NF

- RealNVP: coupling layer

# Outline

## 1. NF examples

Linear normalizing flows

Gaussian autoregressive NF

RealNVP: coupling layer



# Jacobian structure

## Normalizing flows log-likelihood

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

The main challenge is a determinant of the Jacobian matrix.

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

Consider a linear layer  $\mathbf{z} = \mathbf{W}\mathbf{x}$ ,  $\mathbf{W} \in \mathbb{R}^{m \times m}$ .

1. Let  $\mathbf{z}$  be a permutation of  $\mathbf{x}$ .
2. Let  $z_j$  depend only on  $x_j$ .

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

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

# Linear normalizing flows

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

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 normalizing flows

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

## 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.

Decomposition should be done only once in the beginning. Next, we fit decomposed matrices ( $\mathbf{P}/\mathbf{L}/\mathbf{U}$  or  $\mathbf{Q}/\mathbf{R}$ ).

---

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

Hoogeboom E., et al. Emerging convolutions for generative normalizing flows, 2019

# Outline

## 1. NF examples

Linear normalizing flows

**Gaussian autoregressive NF**

RealNVP: coupling layer

# 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

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

## 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 an **invertible** and **differentiable** transformation from  $p(\mathbf{z})$  to  $p(\mathbf{x}|\boldsymbol{\theta})$ .
- ▶ It is an autoregressive (AR) NF with the base distribution  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ !
- ▶ Jacobian of such transformation is triangular!

# Gaussian autoregressive NF

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

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

Generation function  $\mathbf{g}_{\theta}(\mathbf{z})$  is **sequential**.

Inference function  $\mathbf{f}_{\theta}(\mathbf{x})$  is **not sequential**.

## Forward KL for NF

$$KL(\pi||p) = -\mathbb{E}_{\pi(\mathbf{x})} [\log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log |\det(\mathbf{J}_{\mathbf{f}})|] + \text{const}$$

- ▶ We need to be able to compute  $\mathbf{f}_{\theta}(\mathbf{x})$  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  $\mathbf{g}_{\theta}(\mathbf{z}) = \mathbf{f}_{\theta}^{-1}(\mathbf{z})$  until we want to sample from the model.

# Gaussian autoregressive NF

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

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

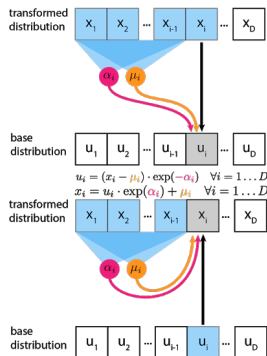
- ▶ Sampling is sequential, density estimation is parallel.
- ▶ Forward KL is a natural loss.

Forward transform:  $\mathbf{f}_{\theta}(\mathbf{x})$

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

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

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



# Outline

## 1. NF examples

Linear normalizing flows

Gaussian autoregressive NF

RealNVP: coupling layer



# RealNVP

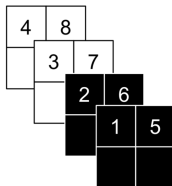
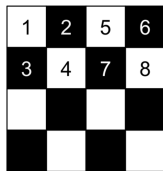
Let split  $\mathbf{x}$  and  $\mathbf{z}$  in two parts:

$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:m}]; \quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] = [\mathbf{z}_{1:d}, \mathbf{z}_{d+1:m}].$$

## Coupling layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \sigma_{\theta}(\mathbf{z}_1) + \mu_{\theta}(\mathbf{z}_1). \end{cases} \quad \begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = (\mathbf{x}_2 - \mu_{\theta}(\mathbf{x}_1)) \odot \frac{1}{\sigma_{\theta}(\mathbf{x}_1)}. \end{cases}$$

## Image partitioning



- ▶ Checkerboard ordering uses masking.
- ▶ Channelwise ordering uses splitting.

# RealNVP

## Coupling layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1; \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \boldsymbol{\sigma}_\theta(\mathbf{z}_1) + \boldsymbol{\mu}_\theta(\mathbf{z}_1). \end{cases} \quad \begin{cases} \mathbf{z}_1 = \mathbf{x}_1; \\ \mathbf{z}_2 = (\mathbf{x}_2 - \boldsymbol{\mu}_\theta(\mathbf{x}_1)) \odot \frac{1}{\boldsymbol{\sigma}_\theta(\mathbf{x}_1)}. \end{cases}$$

Estimating the density takes 1 pass, sampling takes 1 pass!

## Jacobian

$$\det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) = \det \begin{pmatrix} \mathbf{I}_d & 0_{d \times m-d} \\ \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{z}_2}{\partial \mathbf{x}_2} \end{pmatrix} = \prod_{j=1}^{m-d} \frac{1}{\sigma_j(\mathbf{x}_1)}.$$

## Gaussian AR NF

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

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

How to get RealNVP coupling layer from gaussian AR NF?

## Glow samples

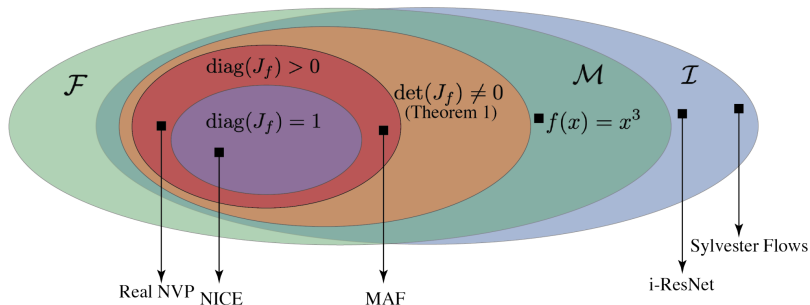
Glow model: coupling layer + linear flows (1x1 convs)



---

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

# Venn diagram for Normalizing flows



- ▶  $\mathcal{I}$  – invertible functions.
- ▶  $\mathcal{F}$  – continuously differentiable functions whose Jacobian is lower triangular.
- ▶  $\mathcal{M}$  – invertible functions from  $\mathcal{F}$ .

# Summary

- ▶ Linear NF try to parametrize set of invertible matrices via matrix decompositions.
- ▶ Gaussian autoregressive NF is an autoregressive model with triangular Jacobian. It has fast inference function and slow generation function. Forward KL is a natural loss function.
- ▶ The RealNVP coupling layer is an effective type of NF (special case of AR NF) that has fast inference and generation modes.