

# Deep Generative Models

## Lecture 2

Roman Isachenko



AI Masters

2026, Spring

# Recap of Previous Lecture

We're given **finite** number of i.i.d. samples  $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$  drawn from an **unknown** distribution  $p_{\text{data}}(\mathbf{x})$ .

## Objective

Our aim is to learn a distribution  $p_{\text{data}}(\mathbf{x})$  that allows us to:

- ▶ Generate new samples from  $p_{\text{data}}(\mathbf{x})$  (sample  $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ ) — **generation**.
- ▶ Evaluate  $p_{\text{data}}(\mathbf{x})$  on novel data (answering “How likely is an object  $\mathbf{x}$ ?”) — **density estimation**;

## Divergence Minimization Task

- ▶  $D(\pi \| p) \geq 0$  for all  $\pi, p \in \mathcal{P}$ ;
- ▶  $D(\pi \| p) = 0$  if and only if  $\pi \equiv p$ .

$$\min_{\theta} D(p_{\text{data}} \| p_{\theta})$$

# Recap of Previous Lecture

## Forward KL Divergence

$$\text{KL}(p_{\text{data}} \| p_{\theta}) = \int \pi(\mathbf{x}) \log \frac{p_{\text{data}}(\mathbf{x})}{p_{\theta}(\mathbf{x})} d\mathbf{x} \rightarrow \min_{\theta}$$

## Reverse KL Divergence

$$\text{KL}(p_{\theta} \| p_{\text{data}}) = \int p_{\theta}(\mathbf{x}) \log \frac{p_{\theta}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})} d\mathbf{x} \rightarrow \min_{\theta}$$

## Maximum Likelihood Estimation (MLE)

$$\theta^* = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i)$$

Maximum likelihood estimation is equivalent to minimizing the Monte Carlo estimate of the forward KL divergence.

# Recap of Previous Lecture

## Likelihood as Product of Conditionals

Let  $\mathbf{x} = (x_1, \dots, x_m)$ , and define  $\mathbf{x}_{1:j} = (x_1, \dots, x_j)$ . Then,

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad \log p_{\theta}(\mathbf{x}) = \sum_{j=1}^m \log p_{\theta}(x_j | \mathbf{x}_{1:j-1})$$

## MLE for Autoregressive Models

$$\theta^* = \arg \max_{\theta} \sum_{i=1}^n \sum_{j=1}^m \log p_{\theta}(x_{ij} | \mathbf{x}_{i,1:j-1})$$

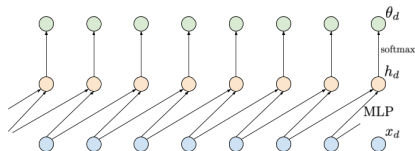
## Sampling

$$\hat{x}_1 \sim p_{\theta}(x_1), \quad \hat{x}_2 \sim p_{\theta}(x_2 | \hat{x}_1), \quad \dots, \quad \hat{x}_m \sim p_{\theta}(x_m | \hat{\mathbf{x}}_{1:m-1})$$

The generated sample is  $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)$ .

# Recap of Previous Lecture

## Autoregressive MLP



## Autoregressive Transformer

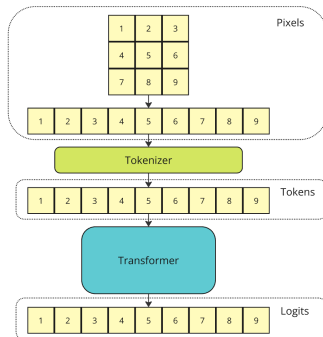
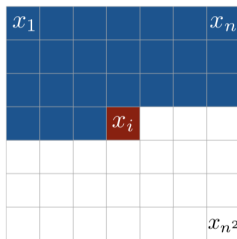


Image credit: [https://jmtomczak.github.io/blog/2/2\\_ARM.html](https://jmtomczak.github.io/blog/2/2_ARM.html)  
Chen M. et al. Generative Pretraining from Pixels, 2020

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

- Linear Normalizing Flows

- Gaussian Autoregressive NF

- Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

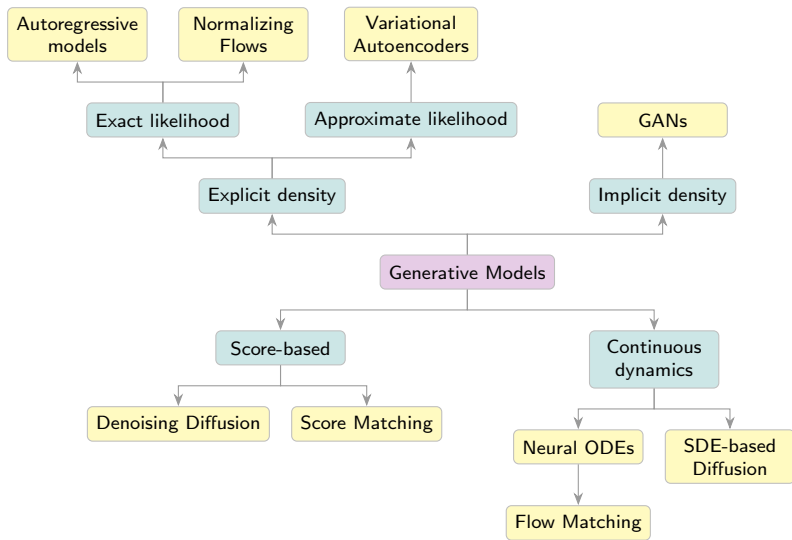
Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)

# Generative Models Taxonomy





# Normalizing Flows: Prerequisites

## Jacobian Matrix

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

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

# Normalizing Flows: Prerequisites

## 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 Variables Theorem (CoV)

Let  $\mathbf{x} \in \mathbb{R}^m$  be a random vector with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a  $C^1$ -diffeomorphism ( $\mathbf{f}$  and  $\mathbf{f}^{-1}$  are continuously differentiable mappings). If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ , then

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

# Normalizing Flows: Prerequisites

## 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 Variables Theorem (CoV)

Let  $\mathbf{x} \in \mathbb{R}^m$  be a random vector with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a  $C^1$ -diffeomorphism ( $\mathbf{f}$  and  $\mathbf{f}^{-1}$  are continuously differentiable mappings). If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ , then

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

# Normalizing Flows: Prerequisites

## 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 Variables Theorem (CoV)

Let  $\mathbf{x} \in \mathbb{R}^m$  be a random vector with density  $p(\mathbf{x})$ , and let  $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$  be a  $C^1$ -diffeomorphism ( $\mathbf{f}$  and  $\mathbf{f}^{-1}$  are continuously differentiable mappings). If  $\mathbf{z} = \mathbf{f}(\mathbf{x})$ , then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J}_{\mathbf{f}})| = p(\mathbf{z}) \left| \det \left( \frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$
$$p(\mathbf{z}) = p(\mathbf{x}) |\det(\mathbf{J}_{\mathbf{f}^{-1}})| = p(\mathbf{x}) \left| \det \left( \frac{\partial \mathbf{x}}{\partial \mathbf{z}} \right) \right| = p(\mathbf{f}^{-1}(\mathbf{z})) \left| \det \left( \frac{\partial \mathbf{f}^{-1}(\mathbf{z})}{\partial \mathbf{z}} \right) \right|$$

# Jacobian Determinant

## Inverse Function Theorem

If the function  $\mathbf{f}$  is invertible and its Jacobian is continuous and non-singular, then

$$\mathbf{J}_{\mathbf{f}^{-1}} = \mathbf{J}_{\mathbf{f}}^{-1};$$

# Jacobian Determinant

## Inverse Function Theorem

If the function  $\mathbf{f}$  is invertible and its Jacobian is continuous and non-singular, then

$$\mathbf{J}_{\mathbf{f}^{-1}} = \mathbf{J}_{\mathbf{f}}^{-1}; \quad |\det(\mathbf{J}_{\mathbf{f}^{-1}})| = \frac{1}{|\det(\mathbf{J}_{\mathbf{f}})|}$$

# Jacobian Determinant

## Inverse Function Theorem

If the function  $\mathbf{f}$  is invertible and its Jacobian is continuous and non-singular, then

$$\mathbf{J}_{\mathbf{f}^{-1}} = \mathbf{J}_{\mathbf{f}}^{-1}; \quad |\det(\mathbf{J}_{\mathbf{f}^{-1}})| = \frac{1}{|\det(\mathbf{J}_{\mathbf{f}})|}$$

- ▶  $\mathbf{x}$  and  $\mathbf{z}$  reside in the same space  $(\mathbb{R}^m)$ .
- ▶  $\mathbf{f}_{\theta}(\mathbf{x})$  is a parameterized transformation.

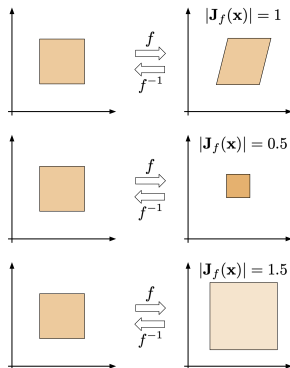
# Jacobian Determinant

## Inverse Function Theorem

If the function  $\mathbf{f}$  is invertible and its Jacobian is continuous and non-singular, then

$$\mathbf{J}_{\mathbf{f}^{-1}} = \mathbf{J}_{\mathbf{f}}^{-1}; \quad |\det(\mathbf{J}_{\mathbf{f}^{-1}})| = \frac{1}{|\det(\mathbf{J}_{\mathbf{f}})|}$$

- ▶  $\mathbf{x}$  and  $\mathbf{z}$  reside in the same space ( $\mathbb{R}^m$ ).
- ▶  $\mathbf{f}_{\theta}(\mathbf{x})$  is a parameterized transformation.
- ▶ The determinant of the Jacobian  $\mathbf{J} = \frac{\partial \mathbf{f}_{\theta}(\mathbf{x})}{\partial \mathbf{x}}$  quantifies how the volume is changed by the transformation.





# Fitting Normalizing Flows

## MLE Problem

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

# Fitting Normalizing Flows

## MLE Problem

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

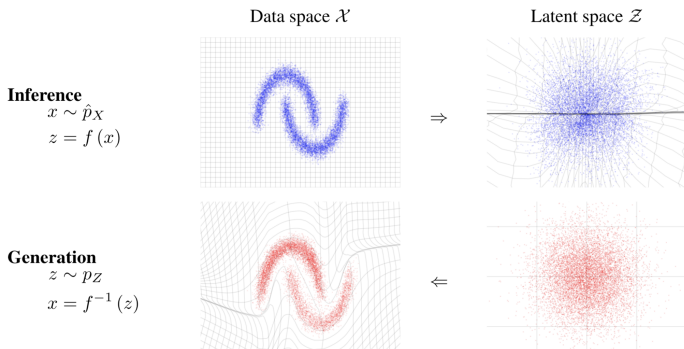
$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log |\det(\mathbf{J}_{\mathbf{f}})| \rightarrow \max_{\theta}$$

# Fitting Normalizing Flows

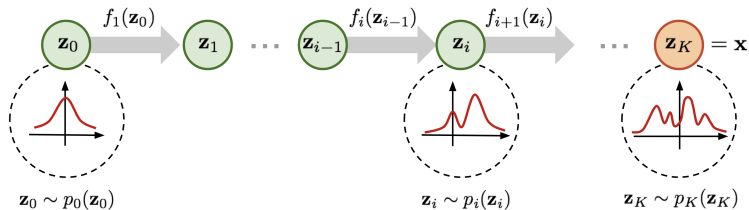
## MLE Problem

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

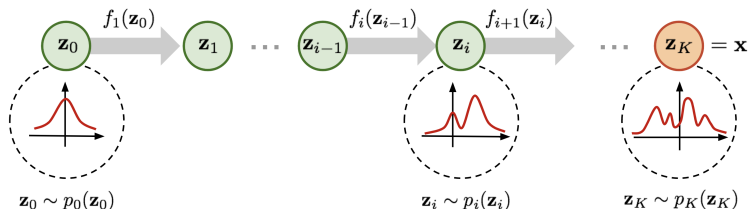
$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_{\theta}(\mathbf{x})) + \log |\det(\mathbf{J}_{\mathbf{f}})| \rightarrow \max_{\theta}$$



# Composition of Normalizing Flows



# Composition of Normalizing Flows

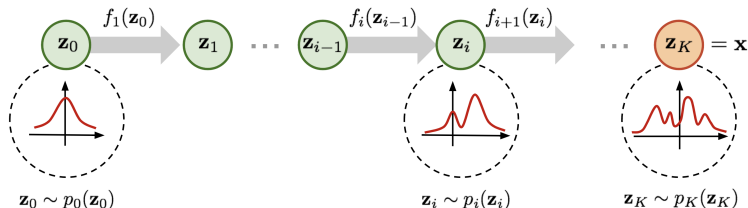


## Theorem

If every  $\{\mathbf{f}_k\}_{k=1}^K$  satisfies the conditions of the change-of-variables theorem, then the composition  $\mathbf{f}(\mathbf{x}) = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})$  also satisfies them.

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

# Composition of Normalizing Flows

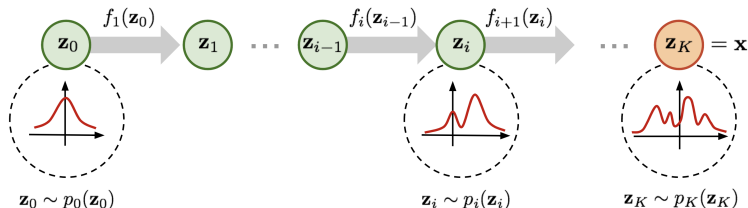


## Theorem

If every  $\{\mathbf{f}_k\}_{k=1}^K$  satisfies the conditions of the change-of-variables theorem, then the composition  $\mathbf{f}(\mathbf{x}) = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})$  also satisfies them.

$$p_{\theta}(\mathbf{x}) = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left( \frac{\partial \mathbf{f}_K}{\partial \mathbf{f}_{K-1}} \dots \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}} \right) \right|$$

# Composition of Normalizing Flows



## Theorem

If every  $\{\mathbf{f}_k\}_{k=1}^K$  satisfies the conditions of the change-of-variables theorem, then the composition  $\mathbf{f}(\mathbf{x}) = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})$  also satisfies them.

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

# Normalizing Flows (NF)

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

## Definition

A normalizing flow is a  $C^1$ -diffeomorphism that transforms data  $\mathbf{x}$  to noise  $\mathbf{z}$ .



# Normalizing Flows (NF)

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

## Definition

A normalizing flow is a  $C^1$ -diffeomorphism that transforms data  $\mathbf{x}$  to noise  $\mathbf{z}$ .

- ▶ **Normalizing** refers to mapping samples from  $p_{\text{data}}(\mathbf{x})$  to a base distribution  $p(\mathbf{z})$ .
- ▶ **Flow** describes the sequence of transformations that maps samples from  $p(\mathbf{z})$  to the target, more complex distribution.

$$\mathbf{z} = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x}); \quad \mathbf{x} = \mathbf{f}_1^{-1} \circ \dots \circ \mathbf{f}_K^{-1}(\mathbf{z})$$

# Normalizing Flows (NF)

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

## Definition

A normalizing flow is a  $C^1$ -diffeomorphism that transforms data  $\mathbf{x}$  to noise  $\mathbf{z}$ .

- ▶ **Normalizing** refers to mapping samples from  $p_{\text{data}}(\mathbf{x})$  to a base distribution  $p(\mathbf{z})$ .
- ▶ **Flow** describes the sequence of transformations that maps samples from  $p(\mathbf{z})$  to the target, more complex distribution.

$$\mathbf{z} = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x}); \quad \mathbf{x} = \mathbf{f}_1^{-1} \circ \dots \circ \mathbf{f}_K^{-1}(\mathbf{z})$$

## Log-Likelihood

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

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

# Normalizing Flows (NF)

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

## Definition

A normalizing flow is a  $C^1$ -diffeomorphism that transforms data  $\mathbf{x}$  to noise  $\mathbf{z}$ .

- ▶ **Normalizing** refers to mapping samples from  $p_{\text{data}}(\mathbf{x})$  to a base distribution  $p(\mathbf{z})$ .
- ▶ **Flow** describes the sequence of transformations that maps samples from  $p(\mathbf{z})$  to the target, more complex distribution.

$$\mathbf{z} = \mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x}); \quad \mathbf{x} = \mathbf{f}_1^{-1} \circ \dots \circ \mathbf{f}_K^{-1}(\mathbf{z})$$

## Log-Likelihood

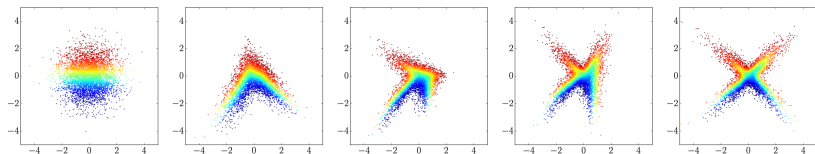
$$\log p_{\theta}(\mathbf{x}) = \log p(\mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^K \log |\det(\mathbf{J}_{\mathbf{f}_k})|$$

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

**Note:** Here we consider only **continuous** random variables.

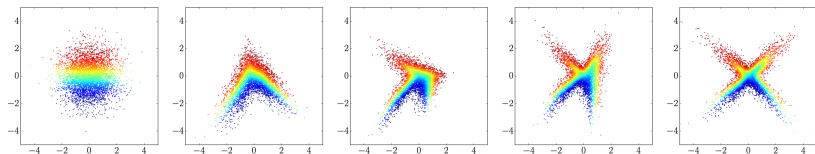
# Normalizing Flows

## Example: 4-Step NF



# Normalizing Flows

## Example: 4-Step NF



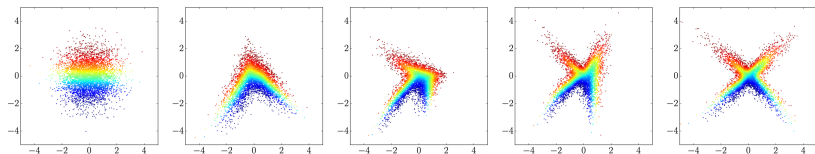
## NF Log-Likelihood

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

What's the computational complexity of evaluating this determinant?

# Normalizing Flows

## Example: 4-Step NF



## NF Log-Likelihood

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

What's the computational complexity of evaluating this determinant?

## Requirements

- ▶ Efficient computation of the Jacobian  $\mathbf{J}_{\mathbf{f}} = \frac{\partial \mathbf{f}_{\theta}(\mathbf{x})}{\partial \mathbf{x}}$
- ▶ Efficient inversion of the transformation  $\mathbf{f}_{\theta}(\mathbf{x})$

---

*Papamakarios G. et al. Normalizing Flows for Probabilistic Modeling and Inference, 2019*

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

- Linear Normalizing Flows

- Gaussian Autoregressive NF

- Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)



# Jacobian Structure

## Normalizing Flows Log-Likelihood

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

The principal computational challenge is evaluating the Jacobian determinant.

# Jacobian Structure

## Normalizing Flows Log-Likelihood

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

The principal computational challenge is evaluating the Jacobian determinant.

### What is $\det(\mathbf{J})$ in These Cases?

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

1.  $\mathbf{z}$  is a permutation of  $\mathbf{x}$ .

# Jacobian Structure

## Normalizing Flows Log-Likelihood

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

The principal computational challenge is evaluating the Jacobian determinant.

### What is $\det(\mathbf{J})$ in These Cases?

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

1.  $\mathbf{z}$  is a permutation of  $\mathbf{x}$ .
2.  $z_j$  depends only on  $x_j$ .

# Jacobian Structure

## Normalizing Flows Log-Likelihood

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

The principal computational challenge is evaluating the Jacobian determinant.

### What is $\det(\mathbf{J})$ in These Cases?

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

1.  $\mathbf{z}$  is a permutation of  $\mathbf{x}$ .
2.  $z_j$  depends only on  $x_j$ .

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

# Jacobian Structure

## Normalizing Flows Log-Likelihood

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

The principal computational challenge is evaluating the Jacobian determinant.

### What is $\det(\mathbf{J})$ in These Cases?

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

1.  $\mathbf{z}$  is a permutation of  $\mathbf{x}$ .
2.  $z_j$  depends only on  $x_j$ .

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

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

# Linear Normalizing Flows

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

In general, matrix inversion has computational complexity  $O(m^3)$ .

# Linear Normalizing Flows

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

In general, matrix inversion has computational complexity  $O(m^3)$ .

## Invertibility

- ▶ Diagonal matrix:  $O(m)$ .
- ▶ Triangular matrix:  $O(m^2)$ .
- ▶ Directly parameterizing all invertible matrices in a continuous way is infeasible  
(there is not surjective function from  $\mathbb{R}^{m^2}$  to the set of all invertible matrices of size  $m \times m$ ).

# Linear Normalizing Flows

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



# Linear Normalizing Flows

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

## 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, and  $\mathbf{U}$  is upper triangular with positive diagonal.

# Linear Normalizing Flows

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

## 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, and  $\mathbf{U}$  is upper triangular with positive diagonal.

### ► QR Decomposition:

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

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

# Linear Normalizing Flows

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

## 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, and  $\mathbf{U}$  is upper triangular with positive diagonal.

### ► QR Decomposition:

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

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

Decomposition is performed only at initialization; the decomposed matrices ( $\mathbf{P}$ ,  $\mathbf{L}$ ,  $\mathbf{U}$  or  $\mathbf{Q}$ ,  $\mathbf{R}$ ) are optimized during training.

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

Linear Normalizing Flows

**Gaussian Autoregressive NF**

Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)

# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

## Sampling

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

# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

## Sampling

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

## Inverse Transformation

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

## Sampling

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

## Inverse Transformation

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

- This gives an  $C^1$ -**diffeomorphism** from  $p(\mathbf{z})$  to  $p_{\theta}(\mathbf{x})$  (assume that  $\sigma_j \neq 0$ ).



# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

## Sampling

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

## Inverse Transformation

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

- ▶ This gives an  $C^1$ -**diffeomorphism** from  $p(\mathbf{z})$  to  $p_{\theta}(\mathbf{x})$  (assume that  $\sigma_j \neq 0$ ).
- ▶ This model is called an autoregressive (AR) NF with base distribution  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ .

# Gaussian Autoregressive Model

Consider the autoregressive model:

$$p_{\theta}(\mathbf{x}) = \prod_{j=1}^m p_{\theta}(x_j | \mathbf{x}_{1:j-1}), \quad p_{\theta}(x_j | \mathbf{x}_{1:j-1}) = \mathcal{N}(\mu_{j,\theta}(\mathbf{x}_{1:j-1}), \sigma_{j,\theta}^2(\mathbf{x}_{1:j-1}))$$

## Sampling

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

## Inverse Transformation

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

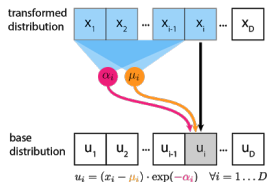
- ▶ This gives an  $C^1$ -**diffeomorphism** from  $p(\mathbf{z})$  to  $p_{\theta}(\mathbf{x})$  (assume that  $\sigma_j \neq 0$ ).
- ▶ This model is called an autoregressive (AR) NF with base distribution  $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$ .
- ▶ The Jacobian matrix of this transformation is triangular.

# Gaussian Autoregressive NF

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

$$\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$$

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$



# Gaussian Autoregressive NF

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

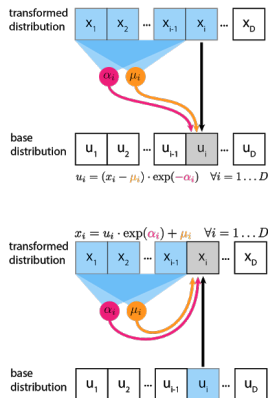
$$\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$$

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

Inverse Transformation:  $\mathbf{f}_\theta^{-1}(\mathbf{z})$

$$\mathbf{x} = \mathbf{f}_\theta^{-1}(\mathbf{z})$$

$$x_j = \sigma_{j,\theta}(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_{j,\theta}(\mathbf{x}_{1:j-1})$$



# Gaussian Autoregressive NF

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

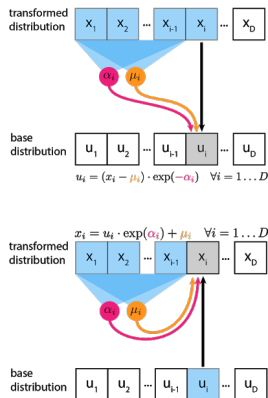
$$\mathbf{z} = \mathbf{f}_\theta(\mathbf{x})$$

$$z_j = \frac{x_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}$$

Inverse Transformation:  $\mathbf{f}_\theta^{-1}(\mathbf{z})$

$$\mathbf{x} = \mathbf{f}_\theta^{-1}(\mathbf{z})$$

$$x_j = \sigma_{j,\theta}(\mathbf{x}_{1:j-1}) \cdot z_j + \mu_{j,\theta}(\mathbf{x}_{1:j-1})$$



- ▶ Sampling must be done sequentially, but density estimation can be parallelized.
- ▶ The forward KL divergence is a natural objective for training.

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)

# RealNVP

Split  $\mathbf{x}$  and  $\mathbf{z}$  into two parts:

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

# RealNVP

Split  $\mathbf{x}$  and  $\mathbf{z}$  into 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

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1 \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \sigma_{\theta}(\mathbf{z}_1) + \mu_{\theta}(\mathbf{z}_1) \end{cases}$$



# RealNVP

Split  $\mathbf{x}$  and  $\mathbf{z}$  into 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

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

# RealNVP

Split  $\mathbf{x}$  and  $\mathbf{z}$  into two parts:

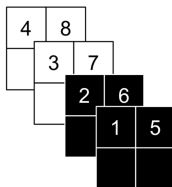
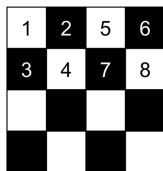
$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:m}]; \quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] = [\mathbf{z}_{1:d}, \mathbf{z}_{d+1:m}]$$

## Coupling Layer

$$\begin{cases} \mathbf{x}_1 = \mathbf{z}_1 \\ \mathbf{x}_2 = \mathbf{z}_2 \odot \sigma_{\theta}(\mathbf{z}_1) + \mu_{\theta}(\mathbf{z}_1) \end{cases}$$

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

## Image Partitioning



- ▶ Checkerboard ordering corresponds to masking.
- ▶ Channelwise ordering relies on splitting.

# RealNVP

## Coupling Layer

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

In both training and sampling, only a single forward pass is needed!

# RealNVP

## Coupling Layer

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

In both training and sampling, only a single forward pass is needed!

## Jacobian

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

# RealNVP

## Coupling Layer

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

In both training and sampling, only a single forward pass is needed!

## Jacobian

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

# RealNVP

## Coupling Layer

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

In both training and sampling, only a single forward pass is needed!

## Jacobian

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

## Gaussian AR NF

$$\begin{aligned} \mathbf{x} = \mathbf{f}_{\theta}^{-1}(\mathbf{z}) &\Rightarrow \mathbf{x}_j = \sigma_{j,\theta}(\mathbf{x}_{1:j-1}) \cdot \mathbf{z}_j + \mu_{j,\theta}(\mathbf{x}_{1:j-1}) \\ \mathbf{z} = \mathbf{f}_{\theta}(\mathbf{x}) &\Rightarrow \mathbf{z}_j = (\mathbf{x}_j - \mu_{j,\theta}(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_{j,\theta}(\mathbf{x}_{1:j-1})}. \end{aligned}$$

How can the RealNVP layer be derived as a special instance of the Gaussian autoregressive NF?

# Outline

## 1. Normalizing Flows (NF)

## 2. NF Examples

Linear Normalizing Flows

Gaussian Autoregressive NF

Coupling Layer (RealNVP)

## 3. Latent Variable Models (LVM)

# Bayesian Framework

## Bayes' Theorem

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

- ▶  $\mathbf{x}$ : observed variables;
- ▶  $\theta$ : unknown latent variables/parameters;
- ▶  $p_{\theta}(\mathbf{x}) = p(\mathbf{x}|\theta)$ : likelihood;
- ▶  $p(\mathbf{x}) = \int p(\mathbf{x}|\theta)p(\theta)d\theta$ : evidence;
- ▶  $p(\theta)$ : prior distribution;
- ▶  $p(\theta|\mathbf{x})$ : posterior distribution.



# Bayesian Framework

## Bayes' Theorem

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

- ▶  $\mathbf{x}$ : observed variables;
- ▶  $\theta$ : unknown latent variables/parameters;
- ▶  $p_{\theta}(\mathbf{x}) = p(\mathbf{x}|\theta)$ : likelihood;
- ▶  $p(\mathbf{x}) = \int p(\mathbf{x}|\theta)p(\theta)d\theta$ : evidence;
- ▶  $p(\theta)$ : prior distribution;
- ▶  $p(\theta|\mathbf{x})$ : posterior distribution.

## Interpretation

- ▶ We begin with unknown variables  $\theta$  and a prior belief  $p(\theta)$ .
- ▶ Once data  $\mathbf{x}$  is observed, the posterior  $p(\theta|\mathbf{x})$  incorporates both prior beliefs and evidence from the data.

# Bayesian Framework

Consider the case where the unobserved variables  $\theta$  are model parameters (i.e.,  $\theta$  are random variables).

- ▶  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ : observed samples;
- ▶  $p(\theta)$ : prior distribution.

# Bayesian Framework

Consider the case where the unobserved variables  $\theta$  are model parameters (i.e.,  $\theta$  are random variables).

- ▶  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ : observed samples;
- ▶  $p(\theta)$ : prior distribution.

## Posterior Distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\theta)p(\theta)}{\int p(\mathbf{X}|\theta)p(\theta)d\theta}$$

# Bayesian Framework

Consider the case where the unobserved variables  $\theta$  are model parameters (i.e.,  $\theta$  are random variables).

- ▶  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ : observed samples;
- ▶  $p(\theta)$ : prior distribution.

## Posterior Distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\theta)p(\theta)}{\int p(\mathbf{X}|\theta)p(\theta)d\theta}$$

If the evidence  $p(\mathbf{X})$  is intractable (due to high-dimensional integration), the posterior cannot be computed exactly.

# Bayesian Framework

Consider the case where the unobserved variables  $\theta$  are model parameters (i.e.,  $\theta$  are random variables).

- ▶  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ : observed samples;
- ▶  $p(\theta)$ : prior distribution.

## Posterior Distribution

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta)p(\theta)}{p(\mathbf{X})} = \frac{p(\mathbf{X}|\theta)p(\theta)}{\int p(\mathbf{X}|\theta)p(\theta)d\theta}$$

If the evidence  $p(\mathbf{X})$  is intractable (due to high-dimensional integration), the posterior cannot be computed exactly.

## Maximum a Posteriori (MAP) Estimation

$$\theta^* = \arg \max_{\theta} p(\theta|\mathbf{X}) = \arg \max_{\theta} (\log p(\mathbf{X}|\theta) + \log p(\theta))$$

# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\theta^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\theta^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

The distribution  $p_{\theta}(\mathbf{x})$  can be highly complex and often intractable (just like the true data distribution  $p_{\text{data}}(\mathbf{x})$ ).

# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\theta^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

The distribution  $p_{\theta}(\mathbf{x})$  can be highly complex and often intractable (just like the true data distribution  $p_{\text{data}}(\mathbf{x})$ ).

## Extended Probabilistic Model

Introduce a latent variable  $\mathbf{z}$  for each observed sample  $\mathbf{x}$ :

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}); \quad \log p_{\theta}(\mathbf{x}, \mathbf{z}) = \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}).$$



# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\theta^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

The distribution  $p_{\theta}(\mathbf{x})$  can be highly complex and often intractable (just like the true data distribution  $p_{\text{data}}(\mathbf{x})$ ).

## Extended Probabilistic Model

Introduce a latent variable  $\mathbf{z}$  for each observed sample  $\mathbf{x}$ :

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}); \quad \log p_{\theta}(\mathbf{x}, \mathbf{z}) = \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}).$$

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

# Latent Variable Models (LVM)

## Maximum Likelihood Estimation (MLE) Problem

$$\theta^* = \arg \max_{\theta} p_{\theta}(\mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^n p_{\theta}(\mathbf{x}_i) = \arg \max_{\theta} \sum_{i=1}^n \log p_{\theta}(\mathbf{x}_i).$$

The distribution  $p_{\theta}(\mathbf{x})$  can be highly complex and often intractable (just like the true data distribution  $p_{\text{data}}(\mathbf{x})$ ).

## Extended Probabilistic Model

Introduce a latent variable  $\mathbf{z}$  for each observed sample  $\mathbf{x}$ :

$$p_{\theta}(\mathbf{x}, \mathbf{z}) = p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z}); \quad \log p_{\theta}(\mathbf{x}, \mathbf{z}) = \log p_{\theta}(\mathbf{x}|\mathbf{z}) + \log p(\mathbf{z}).$$

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

## Motivation

Both  $p_{\theta}(\mathbf{x}|\mathbf{z})$  and  $p(\mathbf{z})$  are usually much simpler than  $p_{\theta}(\mathbf{x})$ .

# Summary

- ▶ The CoV theorem provides a method for computing a random variable's density under an invertible transformation.
- ▶ Normalizing flows transform a simple base distribution into a complex one via a sequence of invertible mappings, each with efficient Jacobian determinants.
- ▶ Linear NFs capture invertible matrices by using matrix decompositions.
- ▶ Gaussian autoregressive NFs are AR models with triangular Jacobians.
- ▶ The RealNVP coupling layer provides an efficient normalizing flow (a special case of AR NF), supporting fast inference and sampling.
- ▶ The Bayesian framework generalizes nearly all standard machine learning methods.