Deep Generative Models

Lecture 6

Roman Isachenko

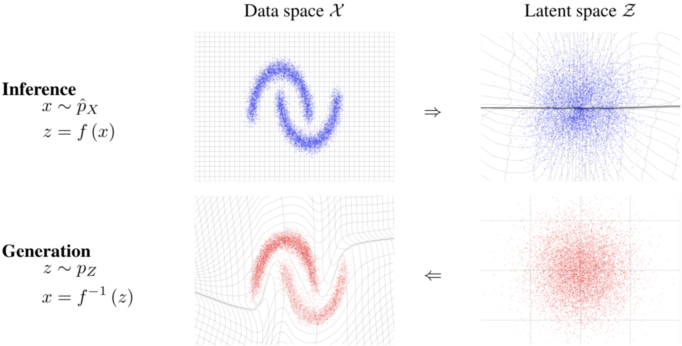
AI Masters

Autumn, 2022

Recap of previous lecture

Definition

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



Log likelihood

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

Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

Recap of previous lecture

Forward KL for flow model

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

Reverse KL for flow model

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

Flow KL duality

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

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

Papamakarios G. et al. Normalizing flows for probabilistic modeling and inference, 2019

Recap of previous lecture

Flow log-likelihood

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

The main challenge is a determinant of the Jacobian.

Linear flows

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

- LU-decomposition
$$\mathbf{W} = \mathbf{P}\mathbf{L}\mathbf{U}.$$
- QR-decomposition
$$\mathbf{W} = \mathbf{Q}\mathbf{R}.$$

Decomposition should be done only once in the beggining. 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. Autoregressive flows
   Gaussian autoregressive flows
   Inverse gaussian autoregressive flows
   RealNVP: coupling layer

2. Normalizing flows as VAE model

Outline

1. Autoregressive flows
   - Gaussian autoregressive flows
   - Inverse gaussian autoregressive flows
   - RealNVP: coupling layer
2. Normalizing flows as VAE model

Outline

1. Autoregressive flows
    - Gaussian autoregressive flows
    - Inverse gaussian autoregressive flows
    - RealNVP: coupling layer

2. Normalizing flows as VAE model

Gaussian autoregressive model

Consider an autoregressive model

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

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 an **invertible** and **differentiable** transformation from $p(\mathbf{z})$ to $p(\mathbf{x}|\boldsymbol{\theta})$.
- It is an autoregressive (AR) flow with the base distribution $p(\mathbf{z}) = \mathcal{N}(0, 1)$!
- Jacobian of such transformation is triangular!

Kingma D. P. et al. Improving Variational Inference with Inverse Autoregressive Flow, 2016

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 NF

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

- 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 model.

Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Outline

Inverse gaussian autoregressive flow (IAF)

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

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 **z** and **x**.

Inverse gaussian autoregressive flow

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

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

Kingma D. P. et al. Improving Variational Inference with Inverse Autoregressive Flow, 2016

Inverse gaussian autoregressive flow (IAF)

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

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

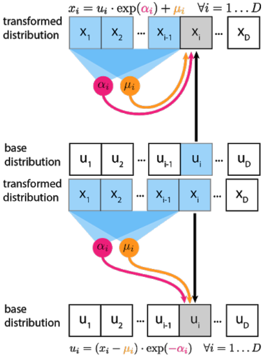
Inverse transform: $g(\mathbf{z}, \boldsymbol{\theta})$

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

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

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

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



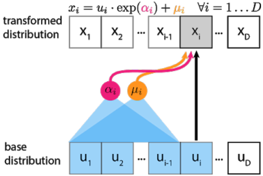


image credit: https://blog.evjang.com/2018/01/nf2.html

Inverse gaussian autoregressive flow (IAF)

Inverse gaussian autoregressive flow

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

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

Reverse KL for NF

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

- 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 $f(\mathbf{x}, \boldsymbol{\theta})$.

Gaussian autoregressive NF

Gaussian AR NF

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

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

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

Inverse gaussian AR NF

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

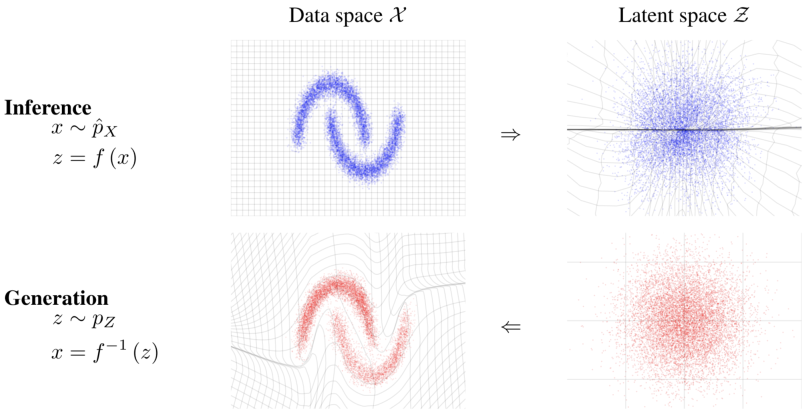
$$\mathbf{z} = f(\mathbf{x}, \boldsymbol{\theta}) \quad \Rightarrow \quad z_j = (x_j - \tilde{\mu}_j(\mathbf{z}_{1:j-1})) \cdot \frac{1}{\tilde{\sigma}_j(\mathbf{z}_{1:j-1})}.$$

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

Kingma D. P. et al. Improving Variational Inference with Inverse Autoregressive Flow, 2016

Autoregressive flows

Gaussian AR NF and inverse gaussian AR NF are mutually interchangeable.



Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

Outline

1. Autoregressive flows
    - Gaussian autoregressive flows
    - Inverse gaussian autoregressive flows
    - RealNVP: coupling layer
2. Normalizing flows as VAE model

RealNVP

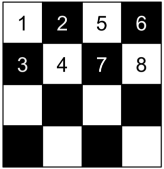
Let split $\mathbf{x}$ and $\mathbf{z}$ in two parts:

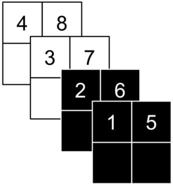
$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:m}]; \quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] = [\mathbf{z}_{1:d}, \mathbf{z}_{d+1:m}].$$

Coupling layer

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

Image partitioning





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

Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

RealNVP

Coupling layer

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

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

Jacobian

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

Gaussian AR NF

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

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

How to get RealNVP coupling layer from gaussian AR NF?

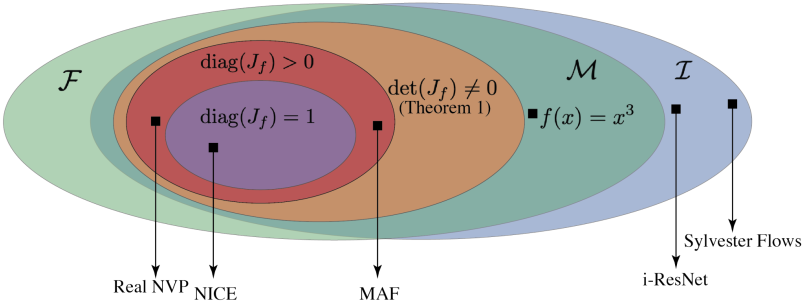
Dinh L., Sohl-Dickstein J., Bengio S. Density estimation using Real NVP, 2016

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 $\mathcal{F}$.

Song Y., Meng C., Ermon S. Mintnet: Building invertible neural networks with masked convolutions, 2019

Outline

VAE vs Normalizing flows

	VAE	**NF**
Objective	ELBO $\mathcal{L}$	Forward KL/MLE
Encoder	stochastic $\mathbf{z} \sim q(\mathbf{z}\|\mathbf{x}, \phi)$	deterministic $\mathbf{z} = f(\mathbf{x}\|\boldsymbol{\theta})$ $q(\mathbf{z}\|\mathbf{x}, \boldsymbol{\theta}) = \delta(\mathbf{z} - f(\mathbf{x}, \boldsymbol{\theta}))$
Decoder	stochastic $\mathbf{x} \sim p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta})$	deterministic $\mathbf{x} = g(\mathbf{z}\|\boldsymbol{\theta})$ $p(\mathbf{x}\|\mathbf{z}, \boldsymbol{\theta}) = \delta(\mathbf{x} - g(\mathbf{z}, \boldsymbol{\theta}))$
Parameters	$\boldsymbol{\phi}, \boldsymbol{\theta}$	$\boldsymbol{\theta} \equiv \boldsymbol{\phi}$

Theorem

MLE for normalizing flow is equivalent to maximization of ELBO for VAE model with deterministic encoder and decoder:

$$p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = \delta(\mathbf{x} - f^{-1}(\mathbf{z}, \boldsymbol{\theta})) = \delta(\mathbf{x} - g(\mathbf{z}, \boldsymbol{\theta}));$$

$$q(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \delta(\mathbf{z} - f(\mathbf{x}, \boldsymbol{\theta})).$$

Nielsen D., et al. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows, 2020

Normalizing flow as VAE

Proof

1. Dirac delta function property

$$\mathbb{E}_{\delta(\mathbf{x}-\mathbf{y})} f(\mathbf{x}) d\mathbf{x} = \int \delta(\mathbf{x}-\mathbf{y}) f(\mathbf{x}) = f(\mathbf{y})$$

2. CoV theorem and Bayes theorem:

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

$$p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) = \frac{p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z})}{p(\mathbf{x}|\boldsymbol{\theta})}; \quad \Rightarrow \quad p(\mathbf{x}|\mathbf{z}, \boldsymbol{\theta}) = p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}) |\det(\mathbf{J}_f)|$$

3. Log-likelihood decomposition

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(\phi, \boldsymbol{\theta}) + KL(q(\mathbf{z}|\mathbf{x}, \phi) || p(\mathbf{z}|\mathbf{x}, \boldsymbol{\theta}))$$

Nielsen D., et al. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows, 2020

Normalizing flow as VAE

Proof

ELBO objective:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} \left[\log p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta}) - \log \frac{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})}{p(\mathbf{z})} \right]$$
$$= \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} \left[\log \frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})} + \log p(\mathbf{z}) \right].$$

1. Dirac delta function property:
$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \log p(\mathbf{z}) = \int \delta(\mathbf{z} - f(\mathbf{x})) p(\mathbf{z}) d\mathbf{z} = \log p(f(\mathbf{x})).$$
2. CoV theorem and Bayes theorem:
$$\mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)} \log \frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x},\phi)} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x},\phi)} \log \frac{p(\mathbf{z}|\mathbf{x})|\det(\mathbf{J}_f)|}{q(\mathbf{z}|\mathbf{x},\phi)} = \log|\det \mathbf{J}_f|.$$
3. Log-likelihood decomposition
$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) = \log p(f(\mathbf{x})) + \log|\det \mathbf{J}_f|.$$

Nielsen D., et al. SurVAE Flows: Surjections to Bridge the Gap between VAEs and Flows, 2020

Summary

- Gaussian autoregressive flow is an autoregressive model with triangular Jacobian. It has fast inference function and slow generation function.
- Inverse gaussian autoregressive flow generate new samples fast, but the inference is slow.
- The RealNVP coupling layer is an effective type of flow (special case of AR flows) that has fast inference and generation modes.
- NF models could be treated as VAE model with deterministic encoder and decoder.