Deep Generative Models

Lecture 2

Roman Isachenko

AI Masters

2025, Autumn

Recap of previous lecture

We are given i.i.d. samples $\{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^m$ from unknown distribution $\pi(\mathbf{x})$.

Goal

We would like to learn a distribution $\pi(\mathbf{x})$ for

- evaluating $\pi(\mathbf{x})$ for new samples (how likely to get object $\mathbf{x}$?);
- sampling from $\pi(\mathbf{x})$ (to get new objects $\mathbf{x} \sim \pi(\mathbf{x})$).

Instead of searching true $\pi(\mathbf{x})$ over all probability distributions, learn function approximation $p(\mathbf{x}|\boldsymbol{\theta}) \approx \pi(\mathbf{x})$.

Divergence

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

Divergence minimization task

$$\min_{\boldsymbol{\theta}} D(\pi||p).$$

Recap of previous lecture

Forward KL

$$KL(\pi||p) = \int \pi(\mathbf{x}) \log \frac{\pi(\mathbf{x})}{p(\mathbf{x}|\boldsymbol{\theta})} d\mathbf{x} \rightarrow \min_{\boldsymbol{\theta}}$$

Reverse KL

$$KL(p||\pi) = \int p(\mathbf{x}|\boldsymbol{\theta}) \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x})} d\mathbf{x} \rightarrow \min_{\boldsymbol{\theta}}$$

Maximum likelihood estimation (MLE)

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

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

Recap of previous lecture

Likelihood as product of conditionals

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

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

MLE problem for autoregressive model

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

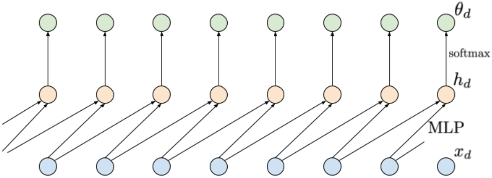
Sampling

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

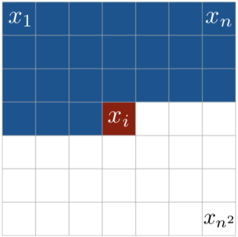
New generated object is $\hat{\mathbf{x}} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_m)$.

Recap of previous lecture

Autoregressive MLP



Autoregressive CNN



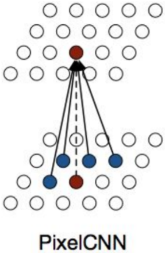


image credit: https://jmtomczak.github.io/blog/2/2_ARM.html

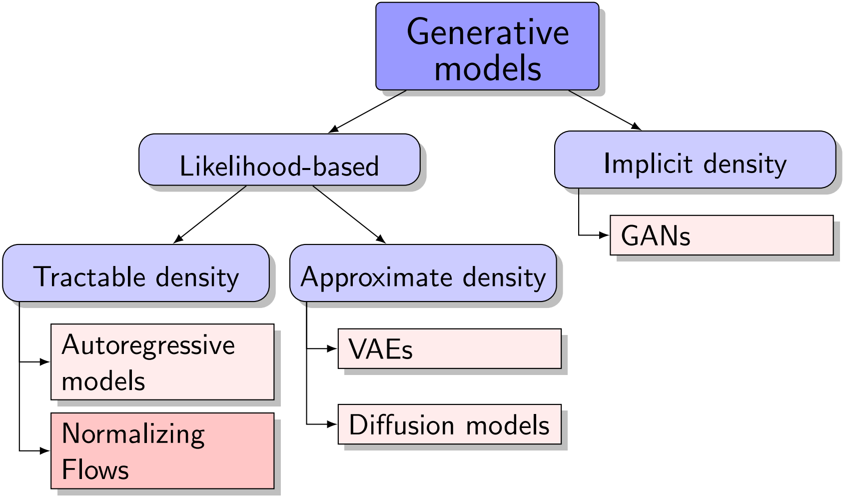
Outline

1. Normalizing flows (NF)

2. NF examples
   - Linear normalizing flows
   - Gaussian autoregressive NF
   - Coupling layer (RealNVP)

Outline

Generative models zoo



Normalizing flows prerequisites

Jacobian matrix

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

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

Change of variable theorem (CoV)

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

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

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

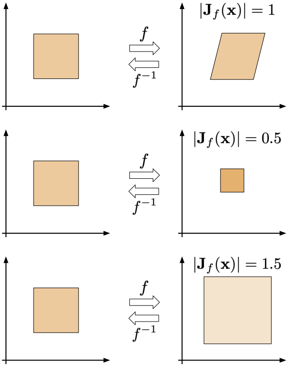
Jacobian determinant

Inverse function theorem

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

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

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



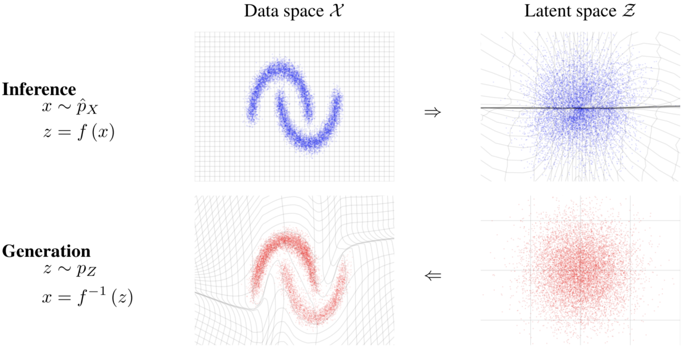
https://jmtomczak.github.io/blog/3/3_flows.html

Fitting normalizing flows

MLE problem

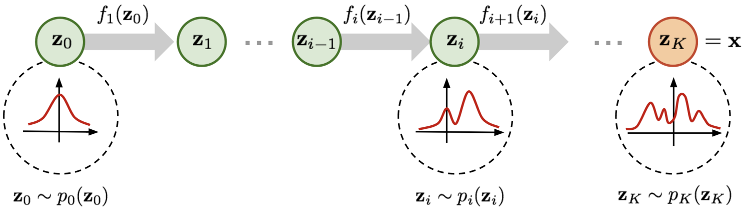
$$p(\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) \left| \det \left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$

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



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

Composition of normalizing flows



Theorem

If $\{\mathbf{f}_k\}_{k=1}^{K}$ satisfy conditions of the change of variable theorem, then $\mathbf{z} = \mathbf{f}(\mathbf{x}) = \mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{x})$ also satisfies it.

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

$$= p(\mathbf{f}(\mathbf{x})) \prod_{k=1}^{K} \left| \det \left(\frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \prod_{k=1}^{K} |\det(\mathbf{J}_{f_k})|$$

https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Normalizing flows (NF)

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

Definition

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

- **Normalizing** means that NF takes samples from $\pi(\mathbf{x})$ and normalizes them into samples from the base 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

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

Log likelihood

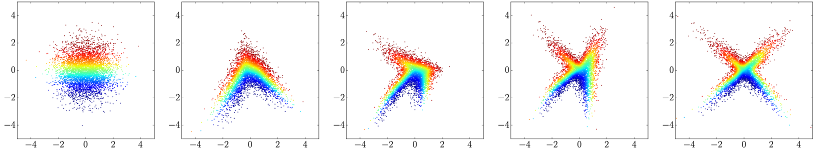
$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^{K} \log|\det(\mathbf{J}_{\mathbf{f}_k})|,$$

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 of a 4-step NF



NF log likelihood

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

What is the complexity of the determinant computation?

What do we need?

- efficient computation of the Jacobian matrix $\mathbf{J}_{\mathbf{f}} = \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})}{\partial \mathbf{x}}$;
- efficient inversion of $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$.

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)

Outline

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

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

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.

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

Linear normalizing flows

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

Matrix decompositions

- **LU-decomposition**

$$\mathbf{W} = \mathbf{PLU},$$

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**

$$\mathbf{W} = \mathbf{QR},$$

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 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

Gaussian autoregressive model

Consider an autoregressive model

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

Sampling

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

Inverse transform

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

- 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!

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

Gaussian autoregressive NF

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

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

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

Forward KL for NF

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

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

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

Gaussian autoregressive NF

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

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

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

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

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

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

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

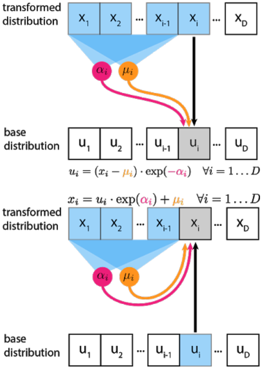


image credit: https://blog.evjang.com/2018/01/nf2.html

Outline

RealNVP

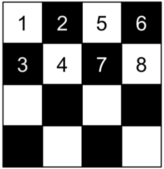
Let split $\mathbf{x}$ and $\mathbf{z}$ in two parts:

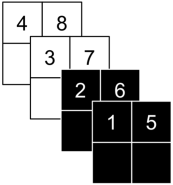
$$\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2] = [\mathbf{x}_{1:d}, \mathbf{x}_{d+1:m}]; \quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2] = [\mathbf{z}_{1:d}, \mathbf{z}_{d+1:m}].$$

Coupling layer

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

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

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

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

Jacobian

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

Gaussian AR NF

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

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

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

Summary

- Change of variable theorem allows to get the density function of the random variable under the invertible transformation.
- Normalizing flows transform a simple base distribution to a complex one via a sequence of invertible transformations with tractable Jacobian.
- Normalizing flows have a tractable likelihood that is given by the change of variable theorem.
- Linear NF try to parametrize set of invertible matrices via matrix decompositions.
- Gaussian autoregressive NF is an autoregressive model with triangular Jacobian.
- The RealNVP coupling layer is an effective type of NF (special case of AR NF) that has fast inference and generation modes.