Deep Generative Models

Lecture 2

Roman Isachenko

AI Masters

2024, Summer

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

Outline

1. Autoregressive models (continued)

2. Normalizing flows (NF)

3. Forward and Reverse KL for NF

4. NF examples
   Linear normalizing flows
   Gaussian autoregressive NF

Outline

1. Autoregressive models (continued)

2. Normalizing flows (NF)

3. Forward and Reverse KL for NF

4. NF examples
   Linear normalizing flows
   Gaussian autoregressive NF

Autoregressive models: MLP

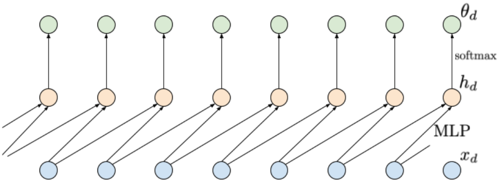
For large j the conditional distribution $p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta})$ could be infeasible. Moreover, the history $\mathbf{x}_{1:j-1}$ has non-fixed length.

Markov assumption

$$p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}) = p(x_j|\mathbf{x}_{j-d:j-1}, \boldsymbol{\theta}), \quad d \text{ is a fixed model parameter.}$$

Example

- $d = 2$;
- $x_j \in \{0, 255\}$;
- $\mathbf{h}_j = \text{MLP}_{\boldsymbol{\theta}}(x_{j-1}, x_{j-2})$;
- $\boldsymbol{\pi}_j = \text{softmax}(\mathbf{h}_j)$;
- $p(x_j|x_{j-1}, x_{j-2}, \boldsymbol{\theta}) = \text{Categorical}(\boldsymbol{\pi}_j)$.



Is it possible to model continuous distributions instead of discrete one?

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

Autoregressive models: PixelCNN

Goal

Model a distribution $\pi(\mathbf{x})$ of natural images.

Solution

Autoregressive model on 2D pixels

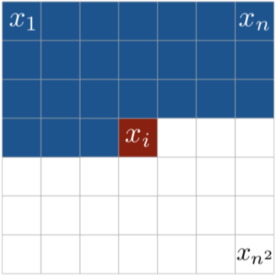
$$p(\mathbf{x}|\boldsymbol{\theta}) = \prod_{j=1}^{\text{width}\times\text{height}} p(x_j|\mathbf{x}_{1:j-1}, \boldsymbol{\theta}).$$

- We need to introduce the ordering of image pixels.
- The convolution should be **masked** to make them causal.
- The image has RGB channels, these dependencies could be addressed.

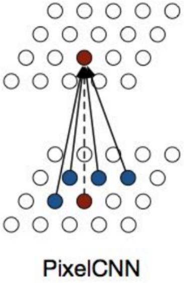
Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks, 2016

Autoregressive models: PixelCNN

Raster ordering

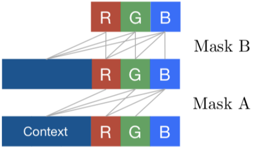


Dependencies between pixels



Mask for the convolution kernel

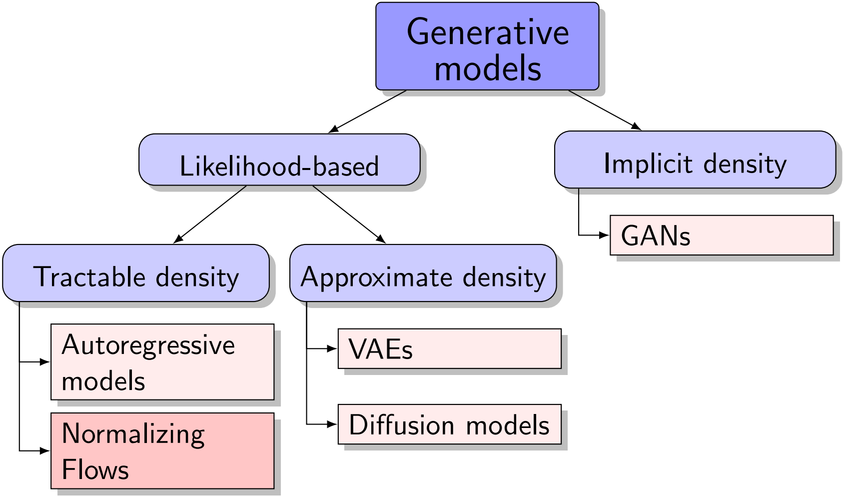
1	1	1
1	0	0
0	0	0



Oord A., Kalchbrenner N., Kavukcuoglu K. Pixel recurrent neural networks, 2016

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

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

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

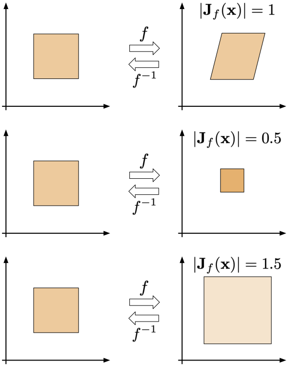
Jacobian determinant

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.



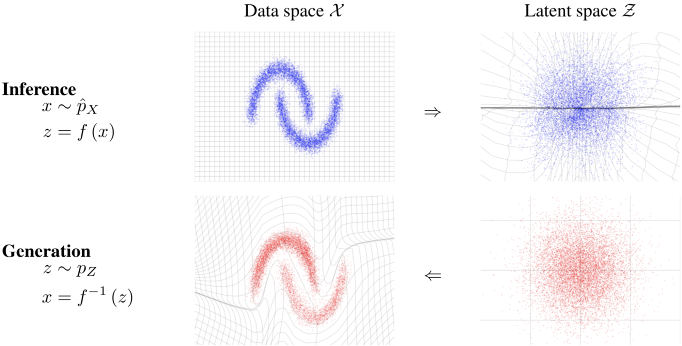
https://jmtomczak.github.io/blog/3/3_flows.html

Fitting normalizing flows

MLE problem

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

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



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

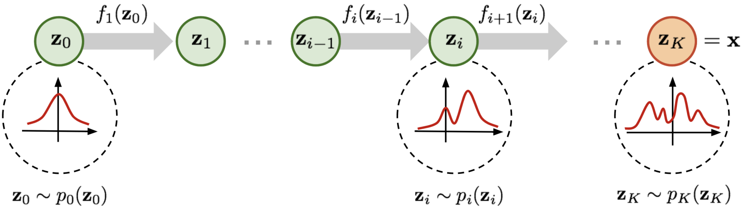
Composition of normalizing flows

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.

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

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



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 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

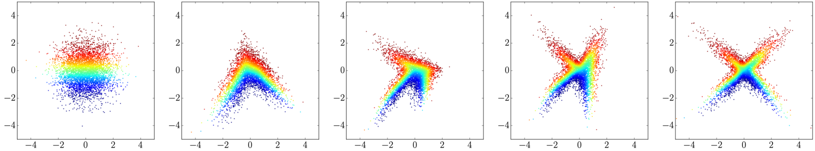
$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{f}_K \circ \cdots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^{K} \log|\det(\mathbf{J}_{\mathbf{f}_k})|,$$

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

Forward KL vs Reverse KL

Forward KL $\equiv$ MLE

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

Forward KL for NF model

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

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

- 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 NF.

Forward KL vs Reverse KL

Reverse KL

$$KL(p||\pi) = \int p(\mathbf{x}|\boldsymbol{\theta}) \log \frac{p(\mathbf{x}|\boldsymbol{\theta})}{\pi(\mathbf{x})} d\mathbf{x}$$
$$= \mathbb{E}_{p(\mathbf{x}|\boldsymbol{\theta})} \left[\log p(\mathbf{x}|\boldsymbol{\theta}) - \log \pi(\mathbf{x}) \right] \to \min_{\boldsymbol{\theta}}$$

Reverse KL for NF model (LOTUS trick)

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{z}) + \log |\det(\mathbf{J_f})| = \log p(\mathbf{z}) - \log |\det(\mathbf{J_g})|$$

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

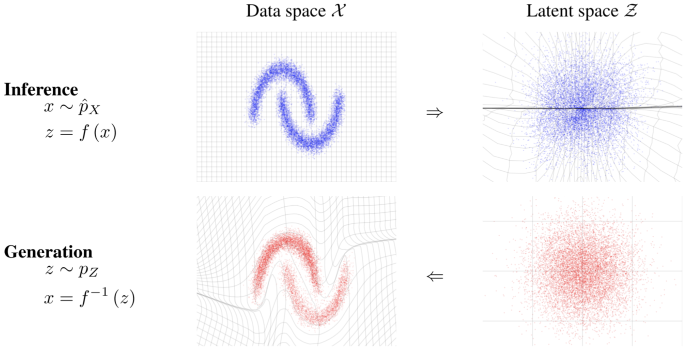
- We need to be able to compute $\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ and its Jacobian.
- We need to be able to sample from the density $p(\mathbf{z})$ (do not need to evaluate it) and to evaluate(!) $\pi(\mathbf{x})$.
- We don't need to think about computing the function $\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$.

Normalizing flows KL duality

Theorem

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

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



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

Normalizing flows KL duality

Theorem

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

Proof

- $\mathbf{z} \sim p(\mathbf{z})$, $\mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$, $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\theta})$;
- $\mathbf{x} \sim \pi(\mathbf{x})$, $\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})$, $\mathbf{z} \sim p(\mathbf{z}|\boldsymbol{\theta})$;

$$\log p(\mathbf{z}|\boldsymbol{\theta}) = \log \pi(\mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})) + \log|\det(\mathbf{J}_\mathbf{g})|;$$
$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{f}_{\boldsymbol{\theta}}(\mathbf{x})) + \log|\det(\mathbf{J}_\mathbf{f})|.$$

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

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

Summary

- PixelCNN model use masked causal convolutions (1D or 2D) to get autoregressive model.
- 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.
- We fit normalizing flows using forward or reverse KL minimization.