Deep Generative Models

Lecture 6

Roman Isachenko

Ozon Masters

Spring, 2022

Recap of previous lecture

Change of variable theorem (CoV)

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

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

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

Inverse function theorem

If function f is invertible and Jacobian is continuous and non-singular, then

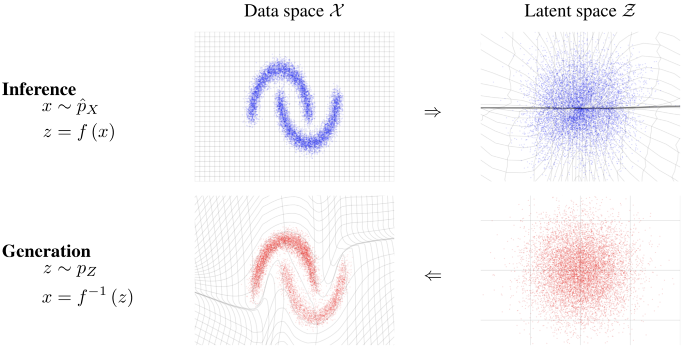
$$\mathbf{J}_f = \mathbf{J}_{g^{-1}} = \mathbf{J}_g^{-1}, \quad |\det(\mathbf{J}_f)| = \frac{1}{|\det(\mathbf{J}_g)|}$$

Recap of previous lecture

MLE problem

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

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



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

Recap of previous lecture

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

Definition

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

- **Normalizing** means that the inverse flow takes samples from $p(\mathbf{x})$ and normalizes them into samples from 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

$$\mathbf{z} = f_K \circ \dots \circ f_1(\mathbf{x}); \quad \mathbf{x} = f_1^{-1} \circ \dots \circ f_K^{-1}(\mathbf{z}) = g_1 \circ \dots \circ g_K(\mathbf{z})$$

Log likelihood

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

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

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.

Residual flows: planar/Sylvester

$$g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{u}\,\sigma(\mathbf{w}^T\mathbf{z} + b); \quad g(\mathbf{z}, \boldsymbol{\theta}) = \mathbf{z} + \mathbf{A}\,\sigma(\mathbf{B}\mathbf{z} + \mathbf{b}).$$

Matrix determinant lemma for calculating the Jacobian.

Linear flows

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

Matrix decompositions (LU or QR helps to parametrize matrix $\mathbf{W}$ and reduce the cost of computing the $det(\mathbf{J})$).

Rezende D. J., Mohamed S. Variational Inference with Normalizing Flows, 2015
Berg R. et al. Sylvester normalizing flows for variational inference, 2018
Kingma D. P., Dhariwal P. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018

Outline

1. Autoregressive flows

2. Inverse autoregressive flows

3. RealNVP: coupling layer

Outline

1. Autoregressive flows
2. Inverse autoregressive flows
3. RealNVP: coupling layer

Gaussian autoregressive model

Consider an autoregressive model

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

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 got an invertible and differentiable transform (it is an autoregressive flow with base distribution $\mathbf{z} = \mathcal{N}(0, 1)$!).

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

Gaussian autoregressive flow

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

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

Forward KL for flow model

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

- 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 flow.

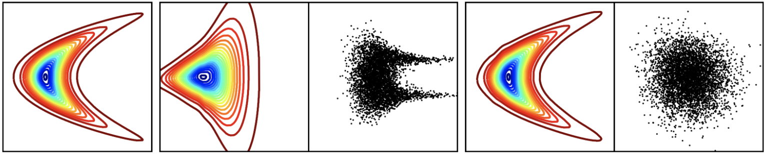
Papamakarios G., Pavlakou T., Murray I. Masked Autoregressive Flow for Density Estimation, 2017

Masked autoregressive flow (MAF)

Gaussian autoregressive model

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

We could use MADE for the conditionals. Samples from the base distribution could be an indicator of how good the flow was fitted.



(a) Target density (b) MADE with Gaussian conditionals (c) MAF with 5 layers

MAF is just a stacked MADE model with different ordering.

- Parallel density estimation.
- Sequential sampling.

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

Autoregressive flows

$$x_j = \tau(z_j, c(\mathbf{z}_{1:j-1})) \quad \Leftrightarrow \quad z_j = \tau^{-1}(x_j, c(\mathbf{z}_{1:j-1}))$$

- $\tau(\cdot, \cdot)$ – coupling law (invertible by first argument, differentiable).
- $c(\cdot)$ – coupling function (do not need to be invertible, could be neural network).

Coupling law $\tau(\cdot, \cdot)$

- $\tau(x, c) = x + c$ – additive;
- $\tau(x, c) = x \odot c_1 + c_2$ – affine.

What is the Jacobian for the additive/affine coupling law?

Jacobian

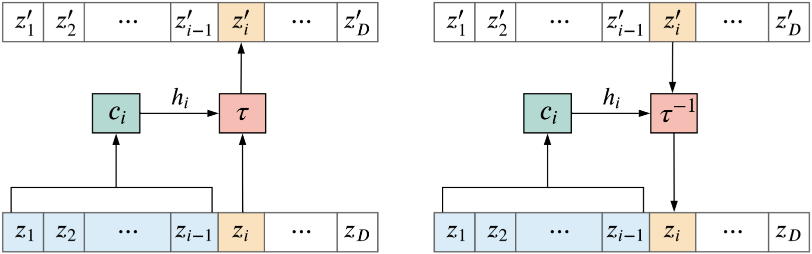
$$\det\left(\frac{\partial \mathbf{x}}{\partial \mathbf{z}}\right) = \prod_{j=1}^{m} \frac{\partial x_j}{\partial z_j} = \prod_{j=1}^{m} \frac{\partial \tau(z_j, c(\mathbf{z}_{1:j-1}))}{\partial z_j}$$

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

Autoregressive flows

Forward and inverse transforms

$$x_j = \tau(z_j, c(\mathbf{z}_{1:j-1})) \quad \Leftrightarrow \quad z_j = \tau^{-1}(x_j, c(\mathbf{z}_{1:j-1}))$$



- Forward transform is **not sequential**.
- Inverse transform is **sequential**.

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

Outline

1. Autoregressive flows

2. Inverse autoregressive flows

3. RealNVP: coupling layer

Inverse 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 $\mathbf{z}$ and $\mathbf{x}$.

Inverse 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 autoregressive flow (IAF)

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

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

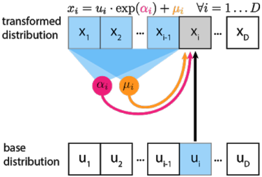
Inverse transform: $g(\mathbf{z}, \boldsymbol{\theta})$

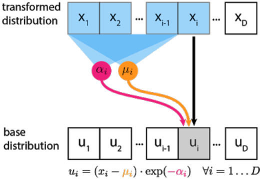
$$z_j = (x_j - \mu_j(\mathbf{x}_{1:j-1})) \cdot \frac{1}{\sigma_j(\mathbf{x}_{1:j-1})};$$

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

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

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





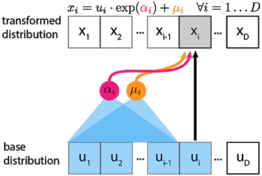


image credit: https://blog.evjang.com/2018/01/nf2.html

Autoregressive flows

Forward and inverse transforms in MAF

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

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

- Sampling is sequential.
- Density estimation is parallel.

Forward and inverse transforms in IAF

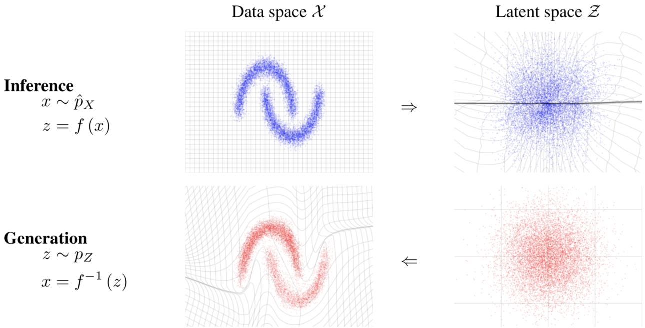
$$\mathbf{x} = g(\mathbf{z}, \boldsymbol{\theta}) \quad \Rightarrow \quad x_j = \tilde{\sigma}_j(\mathbf{z}_{1:j-1}) \cdot z_j + \tilde{\mu}_j(\mathbf{z}_{1:j-1}).$$

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

- Sampling is parallel.
- Density estimation is sequential.

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

Autoregressive flows



- MAF performs parallel inference that is useful for density estimation tasks (forward KL or MLE).
- IAF performs parallel generation that is useful for optimization of reverse KL.

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

Outline

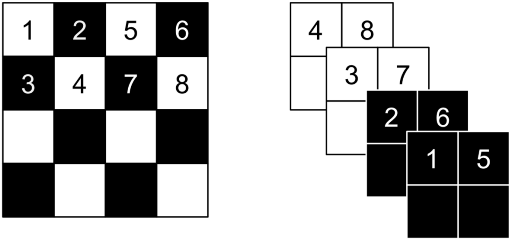
1. Autoregressive flows
2. Inverse autoregressive flows
3. RealNVP: coupling layer

RealNVP

Coupling layer

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \tau(\mathbf{x}_{d:m}, c(\mathbf{x}_{1:d})); \end{cases} \Leftrightarrow \begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = \tau^{-1}(\mathbf{z}_{d:m}, c(\mathbf{z}_{1:d})). \end{cases}$$

Image partitioning



Checkerboard ordering uses masking, channelwise ordering uses splitting.

Dinh L., Krueger D., Bengio Y. NICE: Non-linear Independent Components Estimation, 2014

RealNVP

Affine coupling law

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \mathbf{x}_{d:m} \odot c_1(\mathbf{x}_{1:d}, \boldsymbol{\theta}) + c_2(\mathbf{x}_{1:d}, \boldsymbol{\theta}). \end{cases}$$

$$\begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = (\mathbf{z}_{d:m} - c_2(\mathbf{z}_{1:d}, \boldsymbol{\theta})) \cdot \frac{1}{c_1(\mathbf{z}_{1:d}, \boldsymbol{\theta})}. \end{cases}$$

Jacobian

$$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \det\begin{pmatrix} \mathbf{I}_d & 0_{d \times m-d} \\ \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{1:d}} & \frac{\partial \mathbf{z}_{d:m}}{\partial \mathbf{x}_{d:m}} \end{pmatrix} = \prod_{j=1}^{m-d} c_1(\mathbf{x}_{1:d}, \boldsymbol{\theta})_j.$$

Non-Volume Preserving (the determinant of Jacobian $\neq 1$).

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

MAF vs IAF vs RealNVP

MADE/MAF

$$\mathbf{x} = \boldsymbol{\sigma}(\mathbf{x}) \odot \mathbf{z} + \boldsymbol{\mu}(\mathbf{x}).$$

Estimating the density $p(\mathbf{x}|\boldsymbol{\theta})$ - 1 pass, sampling - m passes.

IAF

$$\mathbf{x} = \tilde{\boldsymbol{\sigma}}(\mathbf{z}) \odot \mathbf{z} + \tilde{\boldsymbol{\mu}}(\mathbf{z}).$$

Estimating the density $p(\mathbf{x}|\boldsymbol{\theta})$ - m passes, sampling - 1 pass.

RealNVP

$$\begin{cases} \mathbf{x}_{1:d} &= \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} &= \mathbf{z}_{d:m} \odot c_1(\mathbf{z}_{1:d}, \boldsymbol{\theta}) + c_2(\mathbf{z}_{1:d}, \boldsymbol{\theta}). \end{cases}$$

Estimating the density $p(\mathbf{x}|\boldsymbol{\theta})$ - 1 pass, sampling - 1 pass.

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

MAF vs IAF vs RealNVP

RealNVP

$$
\begin{cases}
\mathbf{x}_{1:d} & = \mathbf{z}_{1:d}; \\
\mathbf{x}_{d:m} & = \mathbf{z}_{d:m} \odot c_1(\mathbf{z}_{1:d}, \boldsymbol{\theta}) + c_2(\mathbf{z}_{1:d}, \boldsymbol{\theta}).
\end{cases}
$$

- Calculating the density $p(\mathbf{x}|\boldsymbol{\theta})$ - 1 pass.
- Sampling - 1 pass.

RealNVP is a special case of MAF and IAF:

MAF

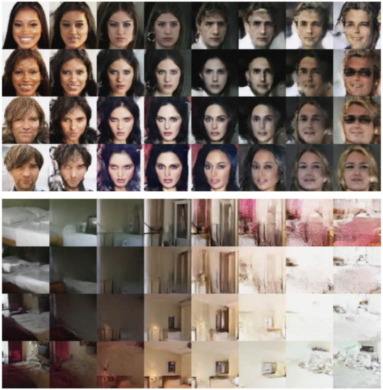
$$
\begin{cases}
\mu_j = 0, \sigma_j = 1, \; j = 1, \dots, d; \\
\mu_j, \sigma_j \text{ – functions of } \mathbf{x}_{1:d}, \; j = d + 1, \dots, m.
\end{cases}
$$

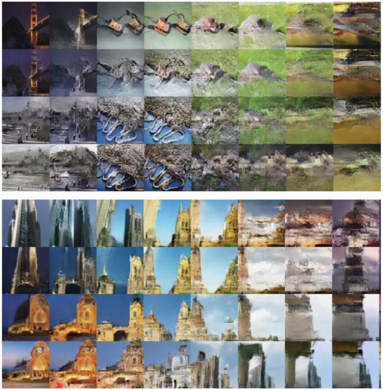
IAF

$$
\begin{cases}
\tilde{\mu}_j = 0, \tilde{\sigma}_j = 1, \; j = 1, \dots, d; \\
\tilde{\mu}_j, \tilde{\sigma}_j \text{ – functions of } \mathbf{z}_{1:d}, \; j = d + 1, \dots, m.
\end{cases}
$$

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

RealNVP samples





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

Linear flows

RealNVP

$$\begin{cases} \mathbf{z}_{1:d} = \mathbf{x}_{1:d}; \\ \mathbf{z}_{d:m} = \tau(\mathbf{x}_{d:m}, c(\mathbf{x}_{1:d})); \end{cases} \quad \Leftrightarrow \quad \begin{cases} \mathbf{x}_{1:d} = \mathbf{z}_{1:d}; \\ \mathbf{x}_{d:m} = \tau^{-1}(\mathbf{z}_{d:m}, c(\mathbf{z}_{1:d})). \end{cases}$$

- First step is a **split** operator which decouples a variable into 2 subparts: $\mathbf{x}_1$ and $\mathbf{x}_2$ (usualy channel-wise).
- We should **permute** components between different layers.

$$\mathbf{z} = \mathbf{W}\mathbf{x}, \quad \mathbf{W} \in \mathbb{R}^{m \times m}$$

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.

Glow samples

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

Summary

- Gaussian autoregressive model is an autoregressive flow with triangular Jacobian.
- Inverse autoregressive flow is able to sample fast, but the inference is slow.
- MAF/IAF is a special case of autoregressive flows.
- The RealNVP is an effective type of flow (special case of AR flows) that uses coupling layer.