Deep Generative Models

7. Normalizing Flow Models

- 국가수리과학연구소 산업수학혁신센터 김민중

Latent Variable Models: Assumption

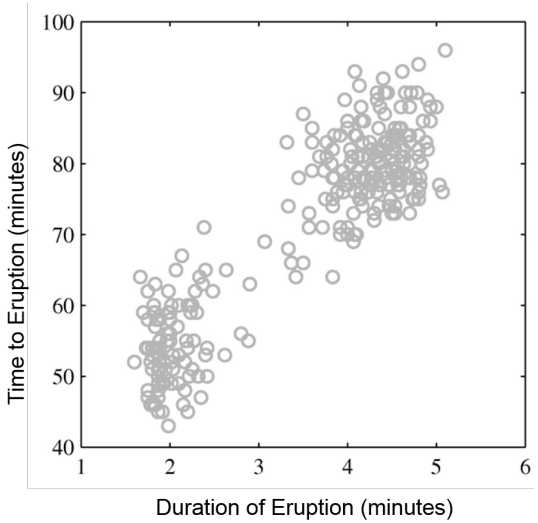
- Observable variables $\boldsymbol{x} \in \mathbb{R}^d$
- Latent variables $\boldsymbol{z} \in \mathbb{R}^h$ (unobservable)

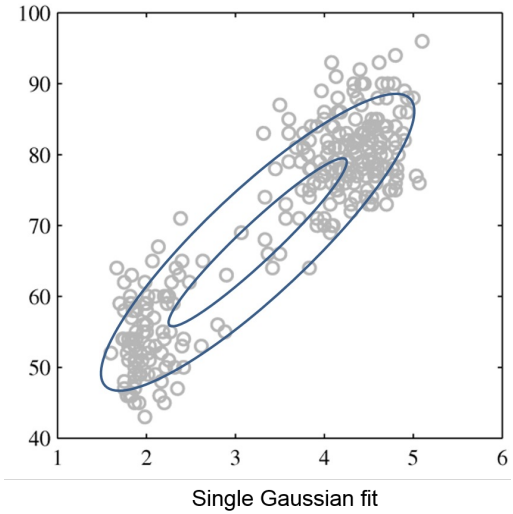
$$p_{data}(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z})$$

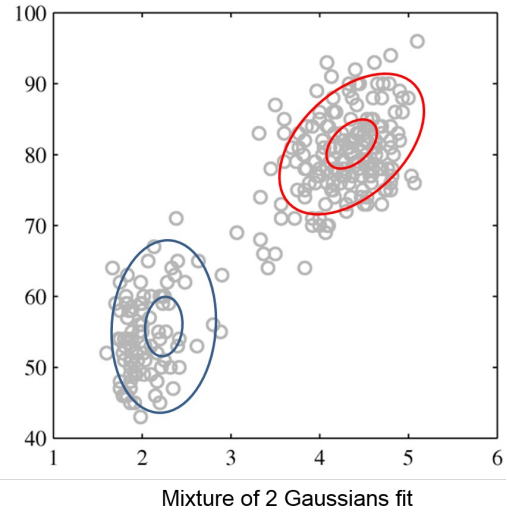
$$or \qquad = \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$$

Example: Mixture of Gaussians

- Mixture of Gaussians. Bayes net: $z \rightarrow \boldsymbol{x}$
  - $z = Categorical(z|\gamma_1, \cdots, \gamma_K)$
  - $p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\mu_k, \Sigma_k)$

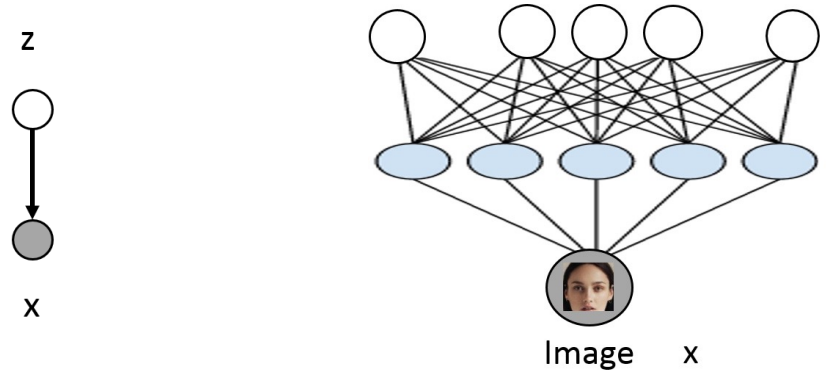






- Clustering: The posterior $p(z|\boldsymbol{x})$ identifies the mixture component

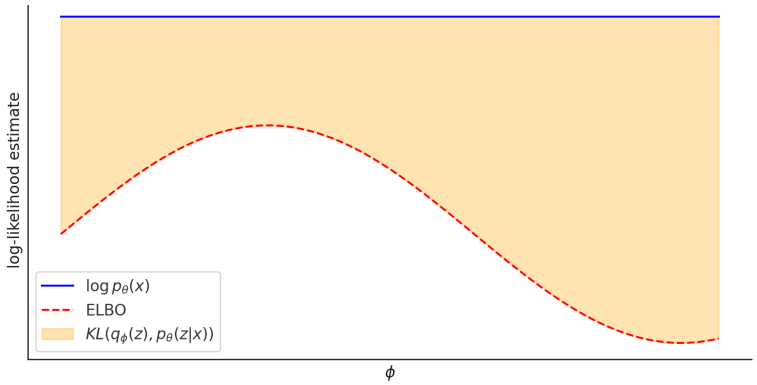
Example: Variational Autoencoder



- A Mixture of an infinite number of Gaussians
  - $\boldsymbol{z} = N(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{z} \in \mathbb{R}^h$
  - $p(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}|\mu_\theta(\boldsymbol{z}), \Sigma_\theta(\boldsymbol{z}))$where $\mu_\theta, \Sigma_\theta$ are neural networks
  - Even though $p(\boldsymbol{x}|\boldsymbol{z})$ is simple, the marginal $p(\boldsymbol{x})$ is very complex/flexible
- Hope that after training, $\boldsymbol{z}$ will correspond to meaningful latent factors of variation (features)
- Unsupervised representation learning
- Features can be computed via $p(\boldsymbol{z}|\boldsymbol{x})$

The Evidence Lower bound

$$\log p_\theta(\boldsymbol{x})$$
$$= E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)$$
$$+ D\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p_\theta(\boldsymbol{z}|\boldsymbol{x})\right)$$



- holds for $\forall q_\phi(\boldsymbol{z}|\boldsymbol{x})$, distribution of $\boldsymbol{z}$ parameterized by ϕ and dependent on $\boldsymbol{x}$. E.g., $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = N\left(\boldsymbol{z}\middle|\boldsymbol{\mu}_\phi(\boldsymbol{x}), \text{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x})\right)\right)$
- We want to jointly optimize over θ and ϕ to maximize the ELBO over a dataset D

Variational Inference

- Variational inference requires that intractable posterior distributions be approximated by a class of known prob. dist.

$$E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)$$

- We need to choose the computationally-feasible approximate posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$
- We need efficient computation of

$$\nabla_\phi E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$$

Stochastic backpropagation

- Reparametrization

$$\boldsymbol{z} = N\left(\boldsymbol{z}|\boldsymbol{\mu}, \mathrm{diag}\left(\boldsymbol{\sigma}^{\boldsymbol{2}}\right)\right) \Leftrightarrow \boldsymbol{z} = \boldsymbol{\mu} + \mathrm{diag}\left(\boldsymbol{\sigma}^{\boldsymbol{2}}\right)\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim N(\boldsymbol{0}, \boldsymbol{I})$$

- Backpropagation with Monte Carlo

$$\nabla_{\phi} E_{\boldsymbol{z}\sim q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}[f_{\theta}(\boldsymbol{z})] \Leftrightarrow E_{\boldsymbol{\epsilon}\sim N(\boldsymbol{\epsilon}|\boldsymbol{0}, \boldsymbol{I})}\left[\nabla_{\phi} f_{\theta}\left(\boldsymbol{\mu}_{\phi} + \mathrm{diag}\left(\boldsymbol{\sigma}_{\phi}^{2}\right)\boldsymbol{\epsilon}\right)\right]$$

Amortized Variational Inference

- **Inference network**: model that learns an inverse map(encoder) from observations to latent variables
- Using this, we can compute a set of global variational parameters ϕ valid for infrence at both training and test time
- The simplest inference models: diagonal Gaussian densities

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) = N\left(\boldsymbol{z}\middle|\boldsymbol{\mu}_\phi(\boldsymbol{x}), \mathrm{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x})\right)\right)$$

VAE: Autoencoder perspective

$$\mathcal{L}(\boldsymbol{x};\theta,\phi) = E_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right) \qquad \text{(ELBO)}$$

1. Take a data point $\boldsymbol{x}'$, map it to sample $\hat{\boldsymbol{z}} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}')$ (encoder)
   - Sample $\hat{\boldsymbol{z}}$ from $q_\phi(\boldsymbol{z}|\boldsymbol{x}') = N\left(\boldsymbol{z}\middle|\boldsymbol{\mu}_\phi(\boldsymbol{x}'), \mathrm{diag}\left(\boldsymbol{\sigma}_\phi^2(\boldsymbol{x}')\right)\right)$, (encoder)
2. Reconstruct $\hat{\boldsymbol{x}}$ by sampling from $p_\theta(\boldsymbol{x}|\hat{\boldsymbol{z}})$ (decoder or generator)
   - $\hat{\boldsymbol{x}} = G_\theta(\hat{\boldsymbol{z}})$
   - The training objective $\mathcal{L}(\boldsymbol{x};\theta,\phi)$
     - Reconstruction error
     - Regularizer

The VAE training algorithm

- Sample minibatch of n training points $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}$ from p_{data}
- Obtain n prob. distributions $q_{\phi}(\boldsymbol{z}|\boldsymbol{x}^{(i)})$ using inference network
- Sample n latent feature points $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(n)}$ from $q_{\phi}(\boldsymbol{z}|\boldsymbol{x}^{(i)})$ resp.
- Update the generator parameters ϕ, θ by stochastic gradient descent

$$\nabla_{\phi}\ell(\theta,\phi) = \frac{1}{n}\nabla_{\phi}\sum_{i=1}^{n}\left[\left\|\boldsymbol{x}^{(i)} - G_{\theta}(\boldsymbol{z}^{(i)})\right\|_2^2 - D\left(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}^{(i)}) \,\|\, p(\boldsymbol{z})\right)\right]$$

$$\nabla_{\theta}\ell(\theta,\phi) = \frac{1}{n}\nabla_{\theta}\sum_{i=1}^{n}\left[\left\|\boldsymbol{x}^{(i)} - G_{\theta}(\boldsymbol{z}^{(i)})\right\|_2^2\right]$$

- Repeat for fixed number of epochs

Summary of Latent Variable Models

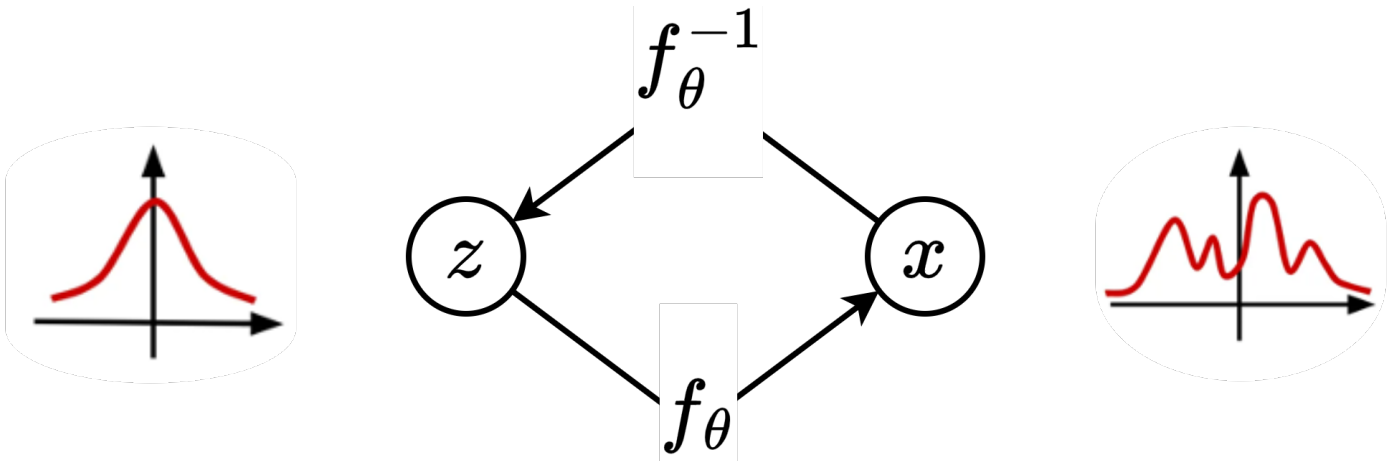
- Combine simple models to get a more flexible one (e.g., mixture of Gaussians)
- Directed model permits ancestral sampling (efficient generation): $\boldsymbol{z} \sim p(\boldsymbol{z})$, $\boldsymbol{x} \sim p_\theta(\boldsymbol{x}|\boldsymbol{z})$
- However, log-likelihood is generally intractable. I.e., learning is difficult
- Joint learning of a model (θ) and an amortized inference component (ϕ) to achieve tractability via ELBO optimization
- Latent representations for any $\boldsymbol{x}$ can be inferred via $q_\phi(\boldsymbol{z}|\boldsymbol{x})$

Recap of likelihood-based learning

- Model families:
  - Autoregressive models: $p_\theta(\boldsymbol{x}) = \prod_{i=1}^{d} p_\theta(x_i|\boldsymbol{x}_{<i})$
  - Latent variable model: $p_\theta(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$
- Autoregressive models provide tractable likelihoods but no direct mechanism for learning features
- Variational autoencoders can learn feature representations (via latent variables $q_\phi(\boldsymbol{z}|\boldsymbol{x})$) but have intractable marginal likelihoods
- Key question: Can we design a latent variable model with **tractable likelihoods**?

Normalizing flow models

- Consider a directed, latent variable model over observed variables X and latent variables Z
- In a normalizing flow model, the mapping between Z and X, given by $\boldsymbol{f}_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$, is deterministic and invertible such that $X = \boldsymbol{f}_\theta(Z)$ and $Z = \boldsymbol{f}_\theta^{-1}(X)$



Continuous random variables (recall)

- If X is a continuous random variable, we can usually represent it using its probability density function $p_X: \mathbb{R} \to \mathbb{R}^+$
- However, we cannot represent this function as a table anymore
- Typically consider parameterized densities:
  - Gaussian: $X = N(X|\mu, \sigma)$ if $p_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$
  - Uniform: $X = U(X|a, b)$ if $p_X(x) = \frac{1}{b-a} 1_{[a \leq x \leq b]}$

- If X is a continuous random vector, we can usually represent it using its joint probability density function:
  - Gaussian: $p_X(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)$

Change of variables formula

- Change of variables(1D case)
  - If $X = f(Z)$ and $f(\cdot)$ is monotone with inverse $Z = f^{-1}(X)$, then

$$p_X(x) = p_Z\left(f^{-1}(x)\right)\left|(f^{-1})'(x)\right|$$

- Note that the ”shape” of $p_X(x)$ is different (more complex) from that of the prior $p_Z(z)$

Change of variables formula

- Change of variables(1D case):
  - If $X = f(Z)$ and $f(\cdot)$ is monotone with inverse $Z = f^{-1}(X)$, then

$$p_X(x) = p_Z\big(f^{-1}(x)\big)|(f^{-1})'(x)|$$

- Proof sketch: Assume $f(\cdot)$ is monotonically increasing

$$F_X(x) = p[X \leq x] = p[f(Z) \leq x] = p[Z \leq f^{-1}(x)] = F_Z\big(f^{-1}(x)\big)$$

- Taking derivatives on both sides:

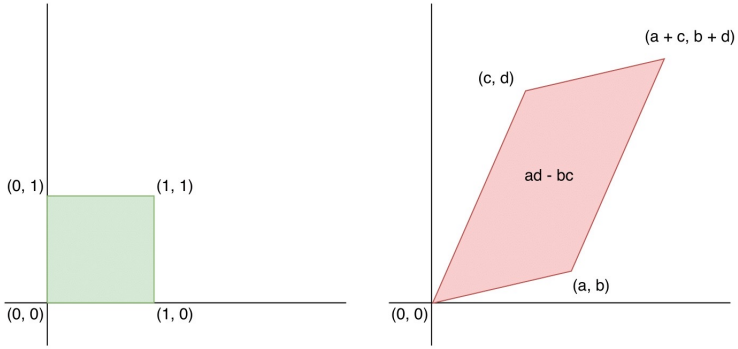
$$p_X(x) = \frac{dF_X(x)}{dx} = \frac{dF_Z\big(f^{-1}(x)\big)}{dx} = p_Z\big(f^{-1}(x)\big)(f^{-1})'(x)$$

- Recall from basic calculus that $[f^{-1}]'(x) = {}^{1}/_{f'(f^{-1}(x))}$
- So, letting $z = f^{-1}(x)$ we can also write

$$p_X(x) = p_Z(z)\left|\frac{1}{f'(z)}\right|$$

Geometry: Determinants and volumes

- Let Z be a uniform random vector in $[0,1]^d$
- Let $X = AZ$ for a square invertible matrix A, with inverse A^{-1}. How is X distributed?
- Geometrically, the matrix A maps the unit hypercube $[0,1]^d$ to a parallelotope
- Hypercube and parallelotope are generalizations of square/cube and parallelogram/parallelopiped to higher dimensions

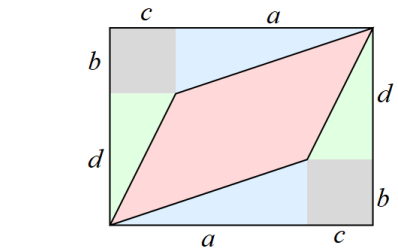


- The matrix $A = \begin{pmatrix} a & c \\ b & d \end{pmatrix}$ maps a unit square to a parallelogram

Geometry: Determinants and volumes

- The volume of the parallelotope is equal to the absolute value of the determinant of the matrix A

$$\det(A) = \det\begin{pmatrix} a & c \\ b & d \end{pmatrix} = ad - bc$$



$$(a+c)(b+d) - ab - 2bc - cd = ad - bc$$

- Let $X = AZ$ for a square invertible matrix A, with inverse A^{-1}
- X is uniformly distributed over the parallelotope of area $|\det(A)|$
- Hence, we have

$$p_X(\boldsymbol{x}) = p_Z(A^{-1}\boldsymbol{x})/|\det(A)| = p_Z(\boldsymbol{z})|\det(A^{-1})|$$

- Note similarity with 1D case formula

Generalized change of variables

- For linear transformations specified via A, change in volume is given by the determinant of A
- For non-linear transformations $\boldsymbol{f}(\cdot)$, the linearized change in volume is given by the determinant of the Jacobian of $\boldsymbol{f}(\cdot)$
- Change of variables (General case): The mapping between Z and X, given by $\boldsymbol{f}: \mathbb{R}^d \rightarrow \mathbb{R}^d$, is invertible such that $X = \boldsymbol{f}(Z)$ and $Z = \boldsymbol{f}^{-1}(X)$

$$p_X(\boldsymbol{x}) = p_Z\left(\boldsymbol{f}^{-1}(\boldsymbol{x})\right)\left|\det\left(\frac{\partial \boldsymbol{f}^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right| = p_Z(\boldsymbol{z})\left|\det\left(\frac{\partial \boldsymbol{f}(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right|^{-1}$$

  - Generalizes the 1D case $p_X(x) = p_Z\left(f^{-1}(x)\right)|(f^{-1})'(x)|$
  - Unlike VAEs, $\boldsymbol{x}, \boldsymbol{z}$ need to be continuous and have the same dimension

Two-Dimensional Example

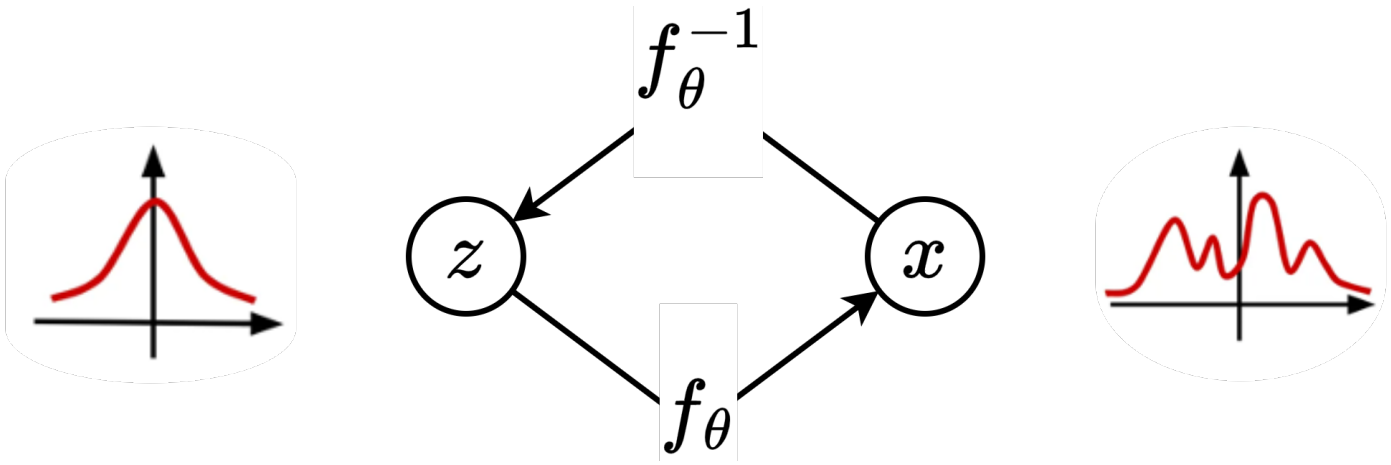
- Let $Z = (Z_1, Z_2)^T$ be continuous random vector with joint density p_Z
- Let $\boldsymbol{u}: \mathbb{R}^2 \to \mathbb{R}^2$ be an invertible transformation denoted $\boldsymbol{u} = (u_1, u_2)$ and let $\boldsymbol{v} = (v_1, v_2)$ be its inverse transformation
- Let $X_1 = u_1(Z_1, Z_2)$ and $X_2 = u_2(Z_1, Z_2)$
- Then, $Z_1 = v_1(X_1, X_2)$ and $Z_2 = v_2(X_1, X_2)$

$$p_X(x_1, x_2) = p_Z\big(v_1(x_1, x_2), v_2(x_1, x_2)\big) \left| \det \begin{pmatrix} \frac{\partial v_1(x_1, x_2)}{\partial x_1} & \frac{\partial v_1(x_1, x_2)}{\partial x_2} \\ \frac{\partial v_2(x_1, x_2)}{\partial x_1} & \frac{\partial v_2(x_1, x_2)}{\partial x_2} \end{pmatrix} \right|$$

$$= p_Z(z_1, z_2) \left| \det \begin{pmatrix} \frac{\partial u_1(z_1, z_2)}{\partial z_1} & \frac{\partial u_1(z_1, z_2)}{\partial z_2} \\ \frac{\partial u_2(z_1, z_2)}{\partial z_1} & \frac{\partial u_2(z_1, z_2)}{\partial z_2} \end{pmatrix} \right|^{-1}$$

Normalizing flow models

- Consider a directed, latent variable model over observed variables X and latent variables Z
- In a normalizing flow model, the mapping between Z and X, given by $\boldsymbol{f}_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$, is deterministic and invertible such that $X = \boldsymbol{f}_\theta(Z)$ and $Z = \boldsymbol{f}_\theta^{-1}(X)$

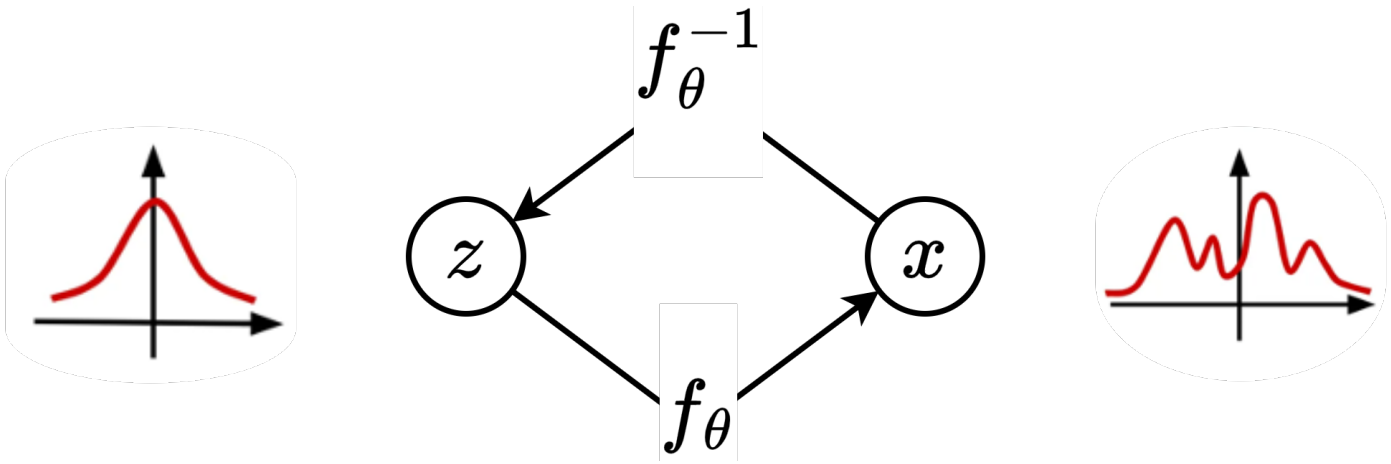


Normalizing flow models

- Using change of variables, the marginal likelihood is given by

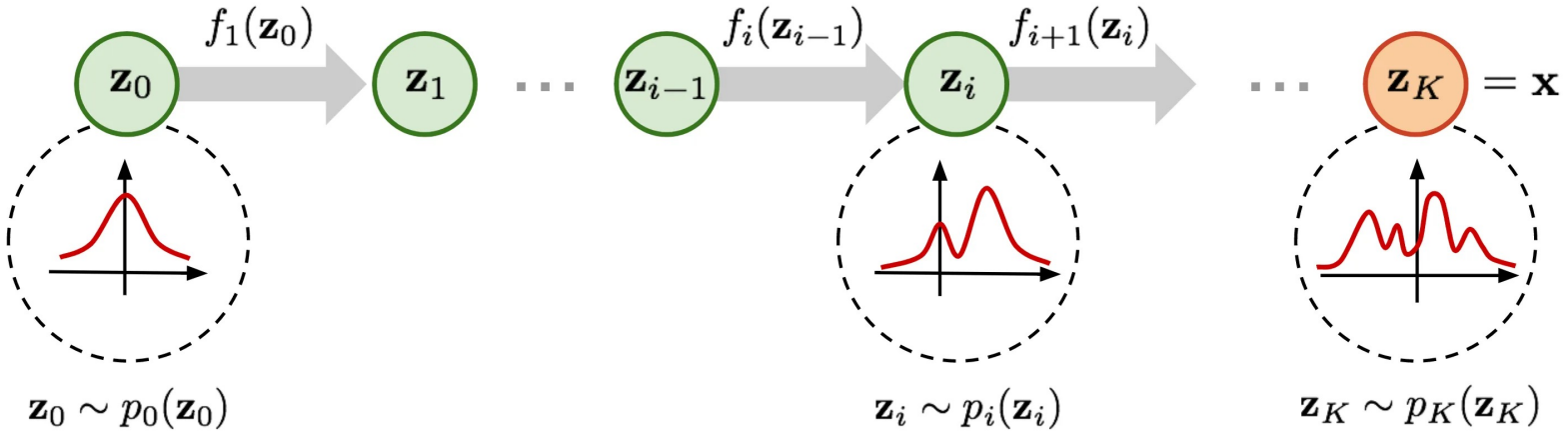
$$p_X(\boldsymbol{x};\theta) = p_Z\left(\boldsymbol{f}_\theta^{-1}(\boldsymbol{x})\right)\left|\det\left(\frac{\partial \boldsymbol{f}_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right| = p_Z(\boldsymbol{z})\left|\det\left(\frac{\partial \boldsymbol{f}_\theta(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right|^{-1}$$

- Note: $\boldsymbol{x}, \boldsymbol{z}$ need to be continuous and have the same dimension



Normalizing Flows

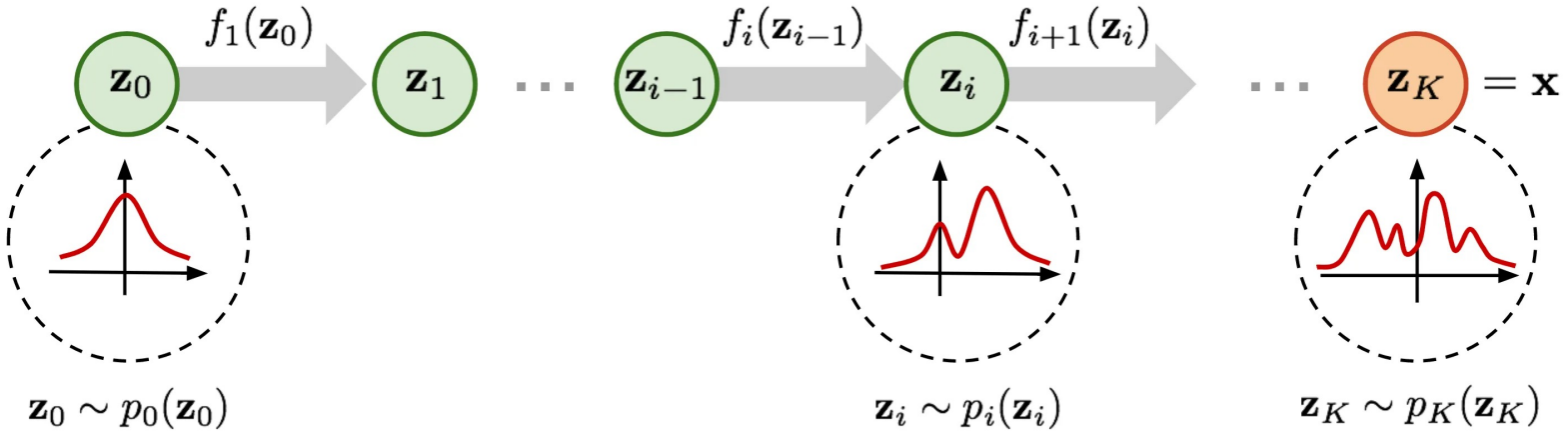
- A normalizing flow describes the **transformation of a probability density through a sequence of invertible mappings**
- By repeatedly applying the rule for change of variables, the initial density ‘flows’ through the sequence of invertible mappings
- Flexible, arbitrarily complex and scalable



A Flow of Transformations

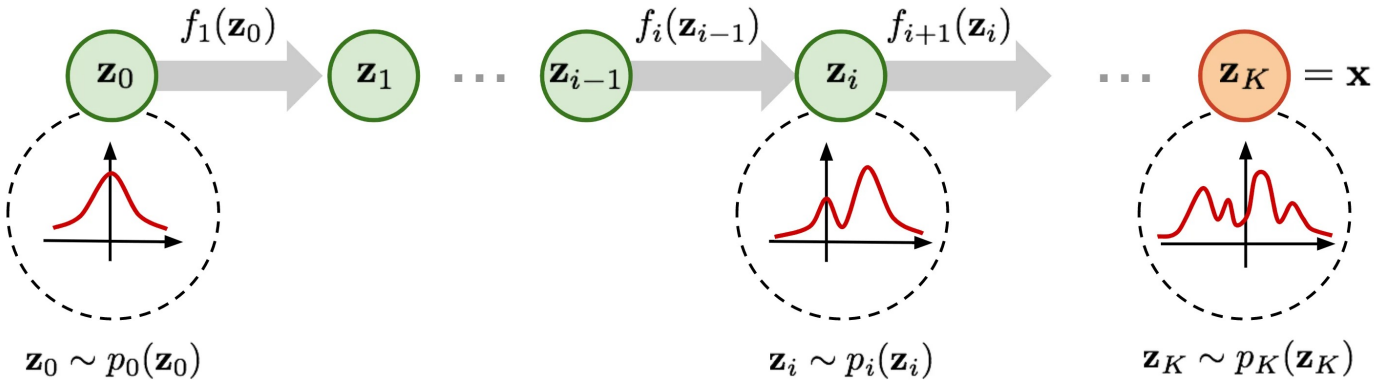
- **Normalizing**: Change of variables gives a normalized density after applying an invertible transformation
- **Flow**: Invertible transformations can be composed with each other

$$\boldsymbol{f}_\theta(\boldsymbol{z}_0) \coloneqq \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_1(\boldsymbol{z}_0) = \boldsymbol{z}_K$$



A Flow of Transformations

$$\boldsymbol{f}_\theta(\boldsymbol{z}_0) \coloneqq \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_1(\boldsymbol{z}_0) = \boldsymbol{z}_K$$



- Start with a simple distribution for $\boldsymbol{z}_0$ (e.g., Gaussian)
- Apply a sequence of K invertible transformations to finally obtain $\boldsymbol{x} = \boldsymbol{z}_K$

$$\boldsymbol{f}_\theta^{-1}(\boldsymbol{x}) = \boldsymbol{f}_1^{-1} \circ \boldsymbol{f}_2^{-1} \circ \cdots \circ \boldsymbol{f}_K^{-1}(\boldsymbol{x})$$

A Flow of Transformations

$$\boldsymbol{f}_\theta(\boldsymbol{z}_0) \coloneqq \boldsymbol{f}_K \circ \boldsymbol{f}_{K-1} \circ \cdots \circ \boldsymbol{f}_1(\boldsymbol{z}_0) = \boldsymbol{z}_K = \boldsymbol{x}$$
$$\boldsymbol{f}_\theta^{-1}(\boldsymbol{x}) = \boldsymbol{f}_1^{-1} \circ \boldsymbol{f}_2^{-1} \circ \cdots \circ \boldsymbol{f}_K^{-1}(\boldsymbol{x})$$

- The marginal likelihood $p_X(\boldsymbol{x})$ is given by

$$p_X(\boldsymbol{x};\theta) = p_Z\left(\boldsymbol{f}_\theta^{-1}(\boldsymbol{x})\right)\left|\det\left(\frac{\partial \boldsymbol{f}_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|$$
$$= p_Z(\boldsymbol{z})\left|\det\left(\frac{\partial \boldsymbol{f}_\theta(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right|^{-1}$$
$$= p_Z\left(\boldsymbol{f}_\theta^{-1}(\boldsymbol{x})\right)\prod_{k=1}^{K}\left|\det\left(\frac{\partial \boldsymbol{f}_k^{-1}(\boldsymbol{x}_k)}{\partial \boldsymbol{x}_k}\right)\right|$$

Learning and Inference

- Learning via maximum likelihood over the dataset D

$$\max_{\theta} \log p_X(D;\theta) = \sum_{\boldsymbol{x}\in D} \log p_Z\left(\boldsymbol{f}_\theta^{-1}(\boldsymbol{x})\right) + \log\left|\det\left(\frac{\partial \boldsymbol{f}_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)\right|$$

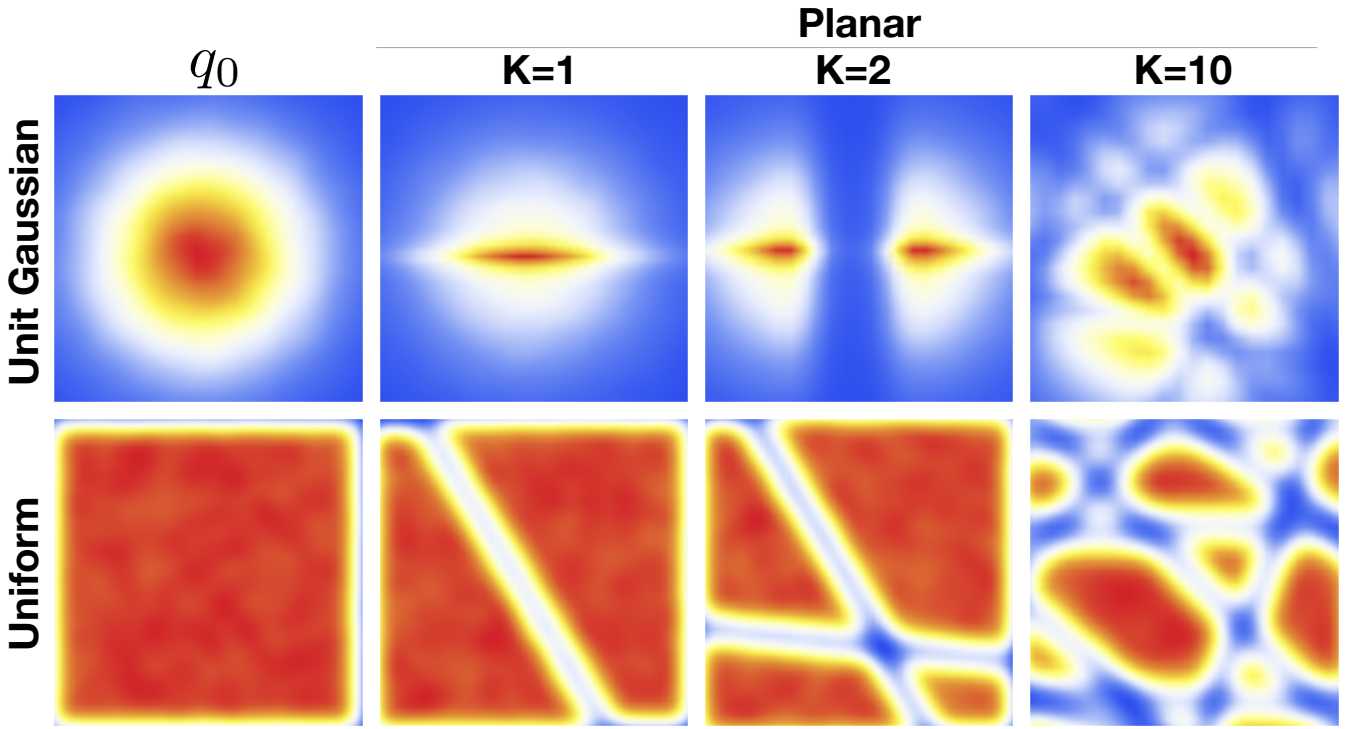
- Exact likelihood evaluation via inverse transformation $\boldsymbol{x} \longmapsto \boldsymbol{z}$ and change of variables formula
- Sampling via forward transformation $\boldsymbol{z} \longmapsto \boldsymbol{x}$

$$\boldsymbol{z} \sim \boldsymbol{p}_Z(\boldsymbol{z}), \boldsymbol{x} = \boldsymbol{f}_\theta(\boldsymbol{z})$$

- Latent representations inferred via inverse transformation (no inference network required): $\boldsymbol{z} = \boldsymbol{f}_\theta^{-1}(\boldsymbol{x})$

Planar flows (Rezende & Mohanmed, 2016)

- Effect of normalizing flow on two distributions
- 10 planar transformations can transform simple distributions into a more complex one



Planar flows (Rezende & Mohanmed, 2016)

- Planar flow

$$\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}h(\boldsymbol{w}^T\boldsymbol{z} + b)$$

- parametrized by $\theta = (\boldsymbol{w}, \boldsymbol{u}, b)$ where $h(\cdot)$ is a nonlinear function
- Absolute value of the determinant of the Jacobian is given by

$$\left|\det\left(\frac{\partial \boldsymbol{f}(\boldsymbol{z})}{\partial \boldsymbol{z}}\right)\right| = |\det(I + h'(\boldsymbol{w}^T\boldsymbol{z} + b)\boldsymbol{u}\boldsymbol{w}^T)|$$
$$= |1 + h'(\boldsymbol{w}^T\boldsymbol{z} + b)\boldsymbol{u}^T\boldsymbol{w}| \quad (\text{matrix determinant lemma})$$

- Need to restrict parameters and non-linearity for the mapping to be invertible.
  - For example, $h = \tanh(\cdot)$ and $h'(\boldsymbol{w}^T\boldsymbol{z} + b)\boldsymbol{u}^T\boldsymbol{w} \geq -1$

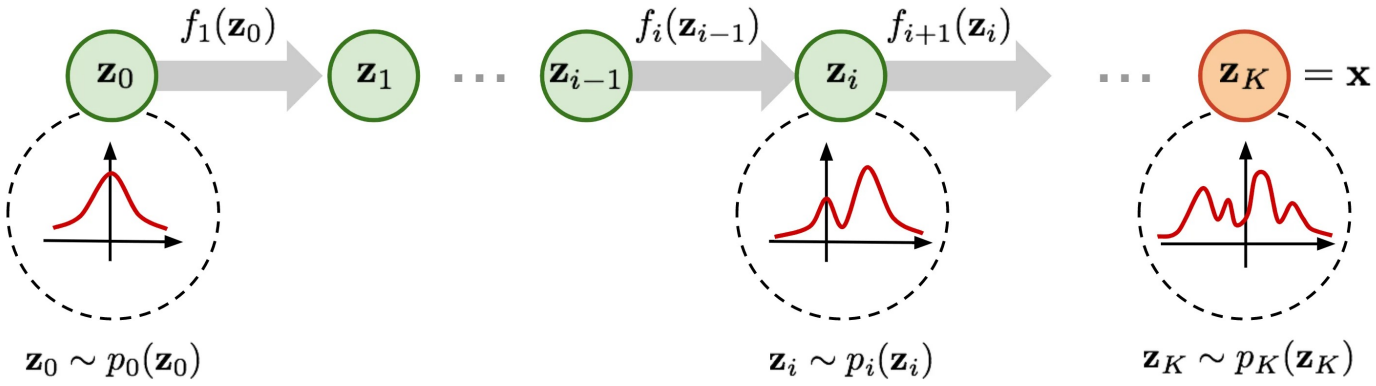
Planar flows (Rezende & Mohanmed, 2016)

- A family of transformations of Planar flows

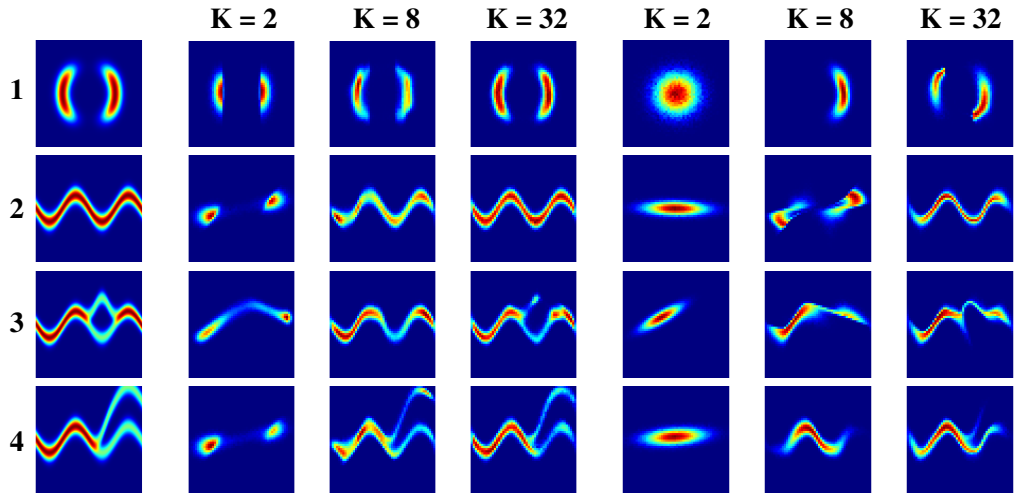
$$\boldsymbol{f}_k(\boldsymbol{z}) = \boldsymbol{z} + \boldsymbol{u}_k h\left(\boldsymbol{w}_k^T \boldsymbol{z} + b_k\right)$$

- parametrized by $(\boldsymbol{w}_k, \boldsymbol{u}_k, b_k)$ where $h(\cdot)$ is a nonlinear function
- Let $\boldsymbol{f}_\theta(\boldsymbol{z}_0) \coloneqq \boldsymbol{f}_K \circ \cdots \boldsymbol{f}_2 \circ \boldsymbol{f}_1(\boldsymbol{z}_0) = \boldsymbol{z}_K = \boldsymbol{x}$
- The log-likelihood $p_X(\boldsymbol{x}; \theta)$ is given by

$$\log p_X(\boldsymbol{x}; \theta) = \log p_Z(\boldsymbol{z}) - \sum_{k=1}^{K} \log\left|1 + h'\left(\boldsymbol{w}_k^T \boldsymbol{z}_{\boldsymbol{k-1}} + b\right)\boldsymbol{u}_k^T \boldsymbol{w}_k\right|$$

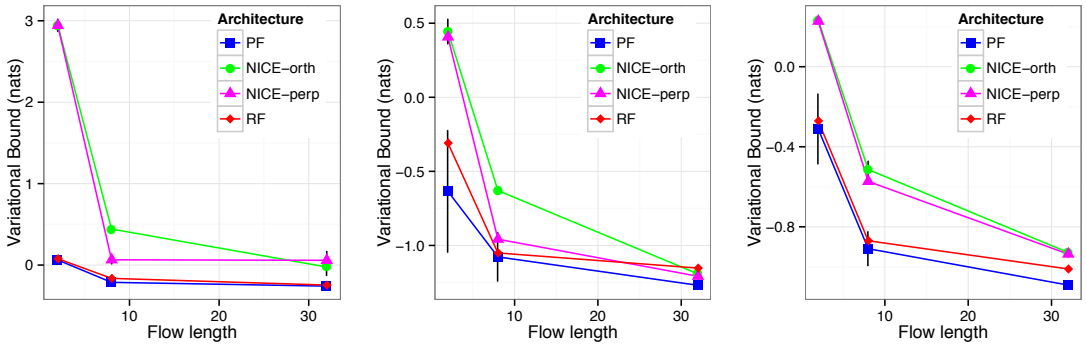


Planar flows (Rezende & Mohanmed, 2016)



(a) (b) Norm. Flow (c) NICE

Approximating four non-Gaussian 2D distributions



(d) Comparison of KL-divergences.

Summary results comparing KL-divergences between the true and approximated densities for the first 3 cases

Desiderata for flow models

- Simple prior $p_Z(\boldsymbol{z})$ that allows for efficient sampling and tractable likelihood evaluation. E.g., isotropic Gaussian
- Invertible transformations with tractable evaluation:
  - Likelihood evaluation requires efficient evaluation of $\boldsymbol{x} \longmapsto \boldsymbol{z}$ mapping
  - Sampling requires efficient evaluation of $\boldsymbol{z} \longmapsto \boldsymbol{x}$ mapping

Desiderata for flow models

- Computing likelihoods also requires the evaluation of determinants $d \times d$ of Jacobian matrices
- Computing the determinant for a $d \times d$ matrix is $O(d^3)$
  - Key idea: Choose transformations so that the resulting Jacobian matrix has special structure
  - For example, the determinant of a triangular matrix is the product of the diagonal entries, i.e., an $O(d)$ operation

Triangular Jacobian

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_d)^T = \boldsymbol{f}(\boldsymbol{z}) = \big(f_1(\boldsymbol{z}), f_2(\boldsymbol{z}), \cdots, f_d(\boldsymbol{z})\big)^T$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} = \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_d} \end{pmatrix}$$

- If $x_i = f_i(\boldsymbol{z})$ only depends on $\boldsymbol{z}_{\leq i}$, then

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} = \begin{pmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial z_1} & \cdots & \frac{\partial f_d}{\partial z_d} \end{pmatrix}$$

- has lower triangular structure

Recap of normalizing flow models

- Transform simple to complex distributions via sequence of invertible transformations
- Directed latent variable models with marginal likelihood given by the change of variables formula
- Triangular Jacobian permits efficient evaluation of log-likelihood
- Examples
  - Invertible transformations with diagonal Jacobians (NICE, Real-NVP)
  - Autoregressive Models as Normalizing Flow Models
  - Invertible CNNs (MintNet)
  - Gaussianization flows

Designing invertible transformations

- Nonlinear Independent Components Estimation (Dinh et al., 2014) composes two kinds of invertible transformations: additive coupling layers and rescaling layers
- Real-NVP (Dinh et al., 2017)
- Inverse Autoregressive Flow (Kingma et al., 2016)
- Masked Autoregressive Flow (Papamakarios et al., 2017)
- I-resnet (Behrmann et al, 2018)
- Glow (Kingma et al, 2018)
- MintNet (Song et al., 2019)
- …

NICE - Additive coupling layers

- Partition the variables $\boldsymbol{z}$ into two disjoint subsets, say $\boldsymbol{z}_{\leq h}$ and $\boldsymbol{z}_{>h}$ for any $1 \leq h < d$
- Forward mapping $\boldsymbol{z} \longmapsto \boldsymbol{x}$:
  - $\boldsymbol{x}_{\leq h} = \boldsymbol{z}_{\leq h}$ (identity transformation)
  - $\boldsymbol{x}_{>h} = \boldsymbol{z}_{>h} + m_\theta(\boldsymbol{z}_{\leq h})$ ($m_\theta(\cdot)$ is a neural network with parameters θ, h input units, and $d - h$ output units)
- Inverse mapping $\boldsymbol{x} \longmapsto \boldsymbol{z}$:
  - $\boldsymbol{z}_{\leq h} = \boldsymbol{x}_{\leq h}$(identity transformation)
  - $\boldsymbol{z}_{>h} = \boldsymbol{x}_{>h} - m_\theta(\boldsymbol{x}_{\leq h})$

NICE - Additive coupling layers

- Jacobian of forward mapping:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} = \begin{pmatrix} I_h & \mathbf{0} \\ \dfrac{\partial \boldsymbol{x}_{>h}}{\partial \boldsymbol{z}_{\leq h}} & I_{d-h} \end{pmatrix}$$

$$\det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right) = 1$$

- Volume preserving transformation since determinant is 1

NICE - Rescaling layers

- Additive coupling layers are composed together (with arbitrary partitions of variables in each layer)
- Final layer of NICE applies a rescaling transformation
- Forward mapping $\boldsymbol{z} \longmapsto \boldsymbol{x}$:

$$x_i = s_i z_i$$

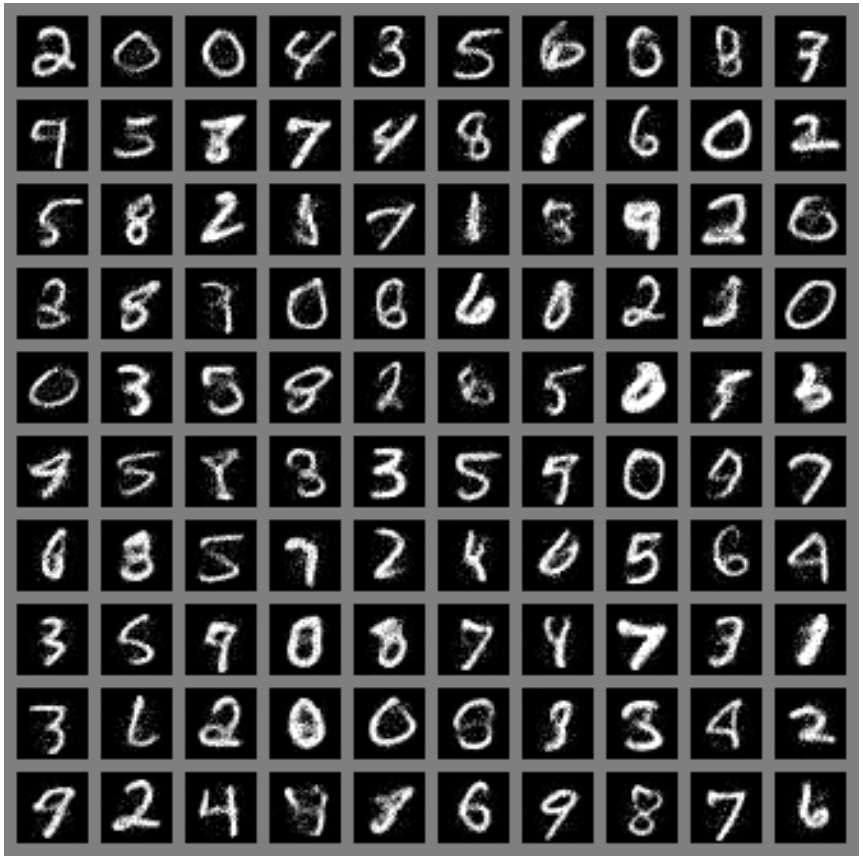
  - where $s_i > 0$ is the scaling factor for the ith dimension
- Inverse mapping $\boldsymbol{x} \longmapsto \boldsymbol{z}$:

$$z_i = \frac{x_i}{s_i}$$

- Jacobian of forward mapping

$$\mathrm{diag}(s_1, s_2, \cdots, s_d)$$

Samples generated via NICE



(a) Model trained on MNIST

(b) Model trained on TFD

Samples generated via NICE

(c) Model trained on SVHN

(d) Model trained on CIFAR-10

Real-NVP: Non-volume preserving extension

- Forward mapping $\boldsymbol{z} \longmapsto \boldsymbol{x}$:
  - $\boldsymbol{x}_{\leq h} = \boldsymbol{z}_{\leq h}$ (identity transformation)
  - $\boldsymbol{x}_{>h} = \boldsymbol{z}_{>h} \odot \exp\big(\alpha_\theta(\boldsymbol{z}_{\leq h})\big) + \mu_\theta(\boldsymbol{z}_{\leq h})$
  - $\alpha_\theta(\cdot)$ and $\mu_\theta(\cdot)$ are both neural networks with parameters θ, h input units, and $d - h$ output units
  - $\odot$ denotes elementwise product
- Inverse mapping $\boldsymbol{x} \longmapsto \boldsymbol{z}$:
  - $\boldsymbol{z}_{\leq h} = \boldsymbol{x}_{\leq h}$ (identity transformation)
  - $\boldsymbol{z}_{>h} = \big(\boldsymbol{x}_{>h} - \mu_\theta(\boldsymbol{x}_{\leq h})\big) \odot \exp\big(-\alpha_\theta(\boldsymbol{x}_{\leq h})\big)$

Real-NVP: Non-volume preserving extension

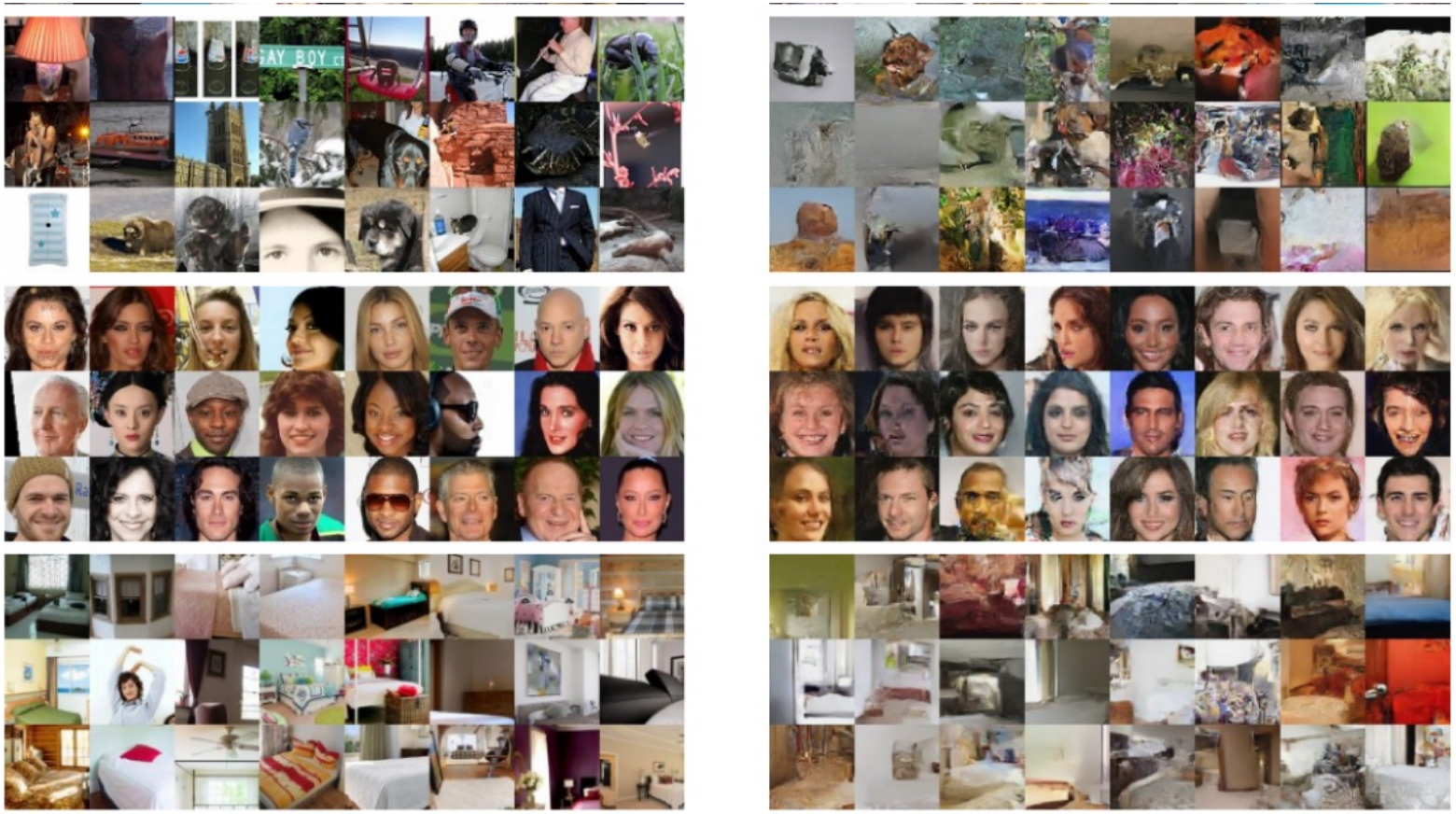
- Jacobian of forward mapping:

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}} = \begin{pmatrix} I_h & \mathbf{0} \\ \dfrac{\partial \boldsymbol{x}_{>h}}{\partial \boldsymbol{z}_{\leq h}} & \operatorname{diag}\left(\exp\left(\alpha_\theta(\boldsymbol{z}_{\leq h})\right)\right) \end{pmatrix}$$

$$\det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}\right) = \prod_{i=h+1}^{d} \exp(\alpha_\theta(\boldsymbol{z}_{\leq h})_i) = \exp\left(\sum_{i=h+1}^{d} \alpha_\theta(\boldsymbol{z}_{\leq h})_i\right)$$

- Non-volume preserving transformation in general since determinant can be less than or greater than 1

Samples generated via Real-NVP



Continuous Autoregressive models as flow models

- Consider a Gaussian autoregressive model:

$$p(\boldsymbol{x}) = p(x_1) \prod_{i=2}^{d} p(x_i|\boldsymbol{x}_{<i})$$

- such that $p(x_i|\boldsymbol{x}_{<i}) = N\left(x_i|\mu_i(\boldsymbol{x}_{<i}), \exp\left(2\alpha_i(\boldsymbol{x}_{<i})\right)\right)$. Here μ_i and α_i are neural networks for $i > 1$ and μ_1 and α_1 are constants

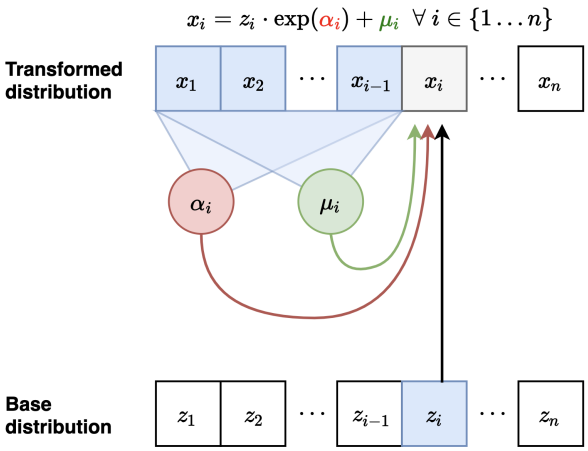
Continuous Autoregressive models as flow models

- Consider a Gaussian autoregressive model:

$$p(\boldsymbol{x}) = p(x_1) \prod_{i=2}^{d} p(x_i|\boldsymbol{x}_{<i})$$

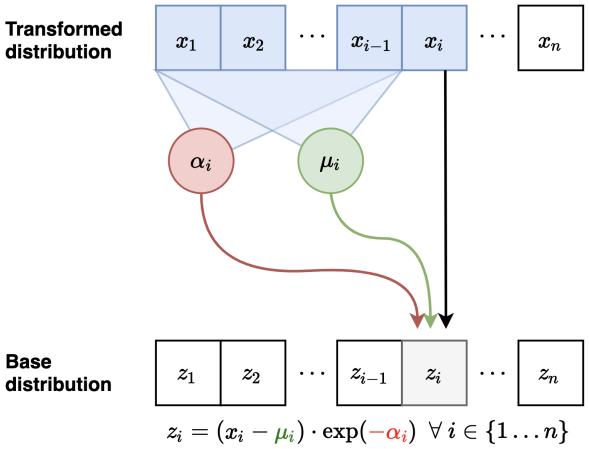
- such that $p(x_i|\boldsymbol{x}_{<i}) = N\left(x_i|\mu_i(\boldsymbol{x}_{<i}), \exp\left(2\alpha_i(\boldsymbol{x}_{<i})\right)\right)$. Here μ_i and α_i are neural networks for $i > 1$ and μ_1 and α_1 are constants
- Sampler for this model:
  - Sample $z_i \sim N(0,1)$ for $i = 1, \cdots, d$
  - Let $x_1 = \exp(\alpha_1)\, z_1 + \mu_1$. Compute $\mu_2(x_1), \alpha_2(x_1)$
  - Let $x_2 = \exp(\alpha_2)\, z_2 + \mu_2$. Compute $\mu_3(x_1, x_2), \alpha_3(x_1, x_2)$
  - Let $x_3 = \exp(\alpha_3)\, z_3 + \mu_3$. $\cdots$
- Flow interpretation: transforms samples from the standard Gaussian $\boldsymbol{z}$ to those generated from the model $\boldsymbol{x}$ via invertible transformations (parameterized by μ_i and α_i)

Masked Autoregressive Flow (MAF)



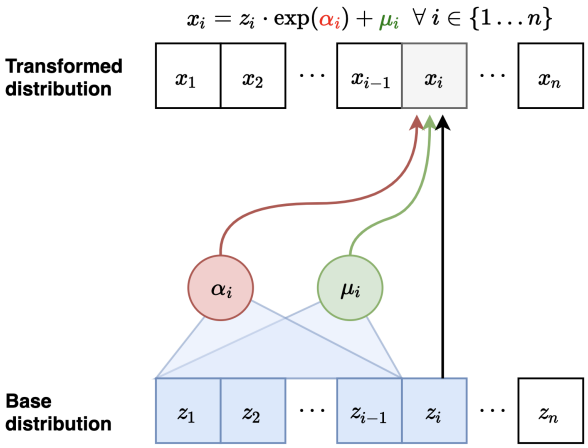
- **Forward** mapping $\boldsymbol{z} \longmapsto \boldsymbol{x}$:
  - Let $x_1 = \exp(\alpha_1)\, z_1 + \mu_1$. Compute $\mu_2(x_1), \alpha_2(x_1)$
  - Let $x_2 = \exp(\alpha_2)\, z_2 + \mu_2$. Compute $\mu_3(x_1, x_2), \alpha_3(x_1, x_2)$
  - Let $x_3 = \exp(\alpha_3)\, z_3 + \mu_3$. $\cdots$
- Sampling is sequential and slow (like autoregressive): $O(d)$ time

Masked Autoregressive Flow (MAF)



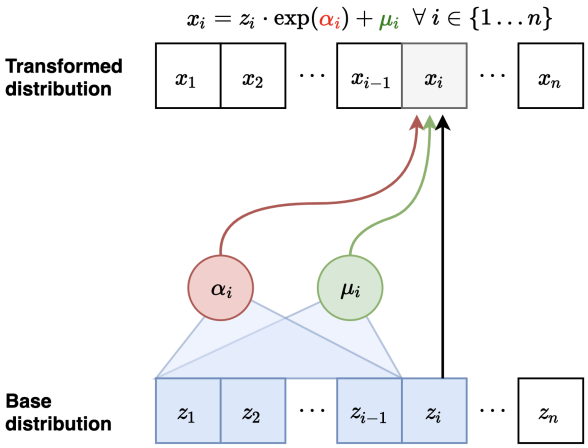
- **Inverse** mapping $\boldsymbol{x} \longmapsto \boldsymbol{z}$:
  - Compute all μ_i, α_i
  - Let $z_1 = (x_1 - \mu_1)/\exp(\alpha_1)$ (scale and shift)
  - Let $z_2 = (x_2 - \mu_2)/\exp(\alpha_2)$
  - Let $z_3 = (x_3 - \mu_3)/\exp(\alpha_3)$. ⋯
- Jacobian is lower diagonal, hence efficient determinant computation
- Likelihood evaluation is easy and parallelizable

Inverse Autoregressive Flow (IAF)



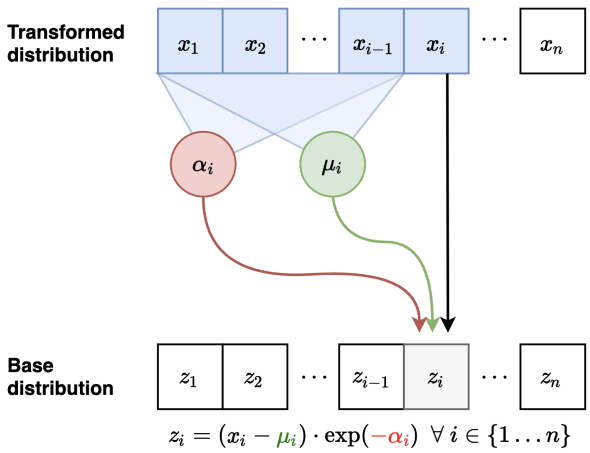
- **Forward** mapping $\boldsymbol{z} \longmapsto \boldsymbol{x}$ (parallel):
  - Sample $z_i \sim N(0,1)$ for $i = 1, \cdots, d$
  - Compute all μ_i, α_i
  - Let $x_1 = \exp(\alpha_1)\, z_1 + \mu_1$
  - Let $x_2 = \exp(\alpha_2)\, z_2 + \mu_2$
  - Let $x_3 = \exp(\alpha_3)\, z_3 + \mu_3 \cdots$

Inverse Autoregressive Flow (IAF)



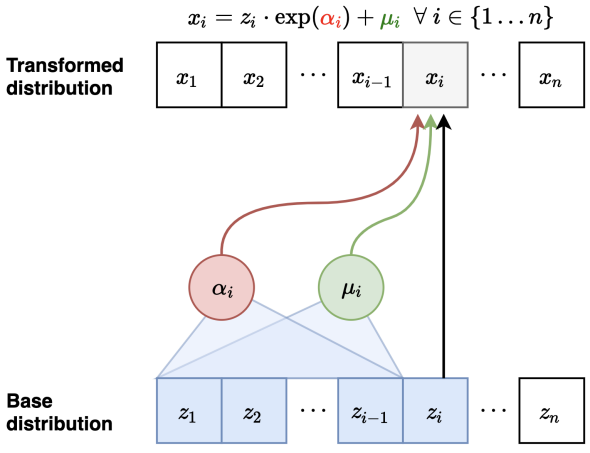
- **Inverse** mapping $\boldsymbol{x} \longmapsto \boldsymbol{z}$ (sequential):
  - Let $z_1 = (x_1 - \mu_1)/\exp(\alpha_1)$. Compute $\mu_2(z_1), \alpha_2(z_1)$
  - Let $z_2 = (x_2 - \mu_2)/\exp(\alpha_2)$. Compute $\mu_3(z_1, z_2), \alpha_3(z_1, z_2)$
  - Let $z_3 = (x_3 - \mu_3)/\exp(\alpha_3)$. $\cdots$
- Fast to sample
- Slow to evaluate likelihoods of data points
- Note: Fast to evaluate likelihoods of a generated point

MAF and IAF



Inverse pass of MAF

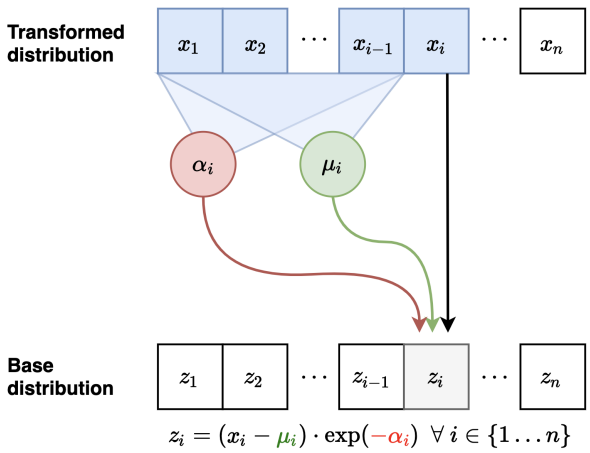
vs.



Forward pass of IAF

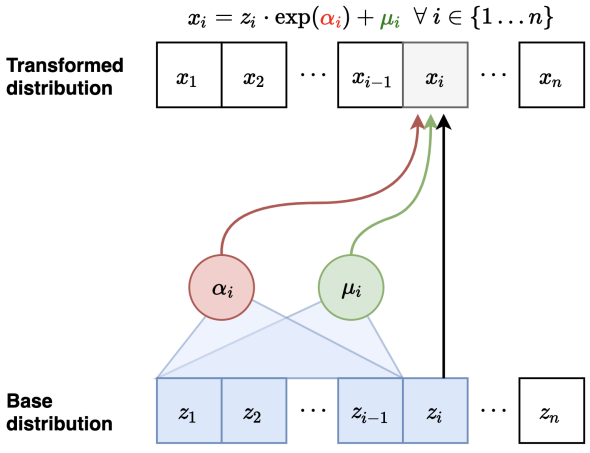
- **IAF is an inverse of MAF**
- Interchanging $\boldsymbol{z}$ and $\boldsymbol{x}$ in the inverse transformation of MAF gives the forward transformation of IAF
- Similarly, forward transformation of MAF is inverse transformation of IAF

MAF and IAF



Inverse pass of MAF

vs.

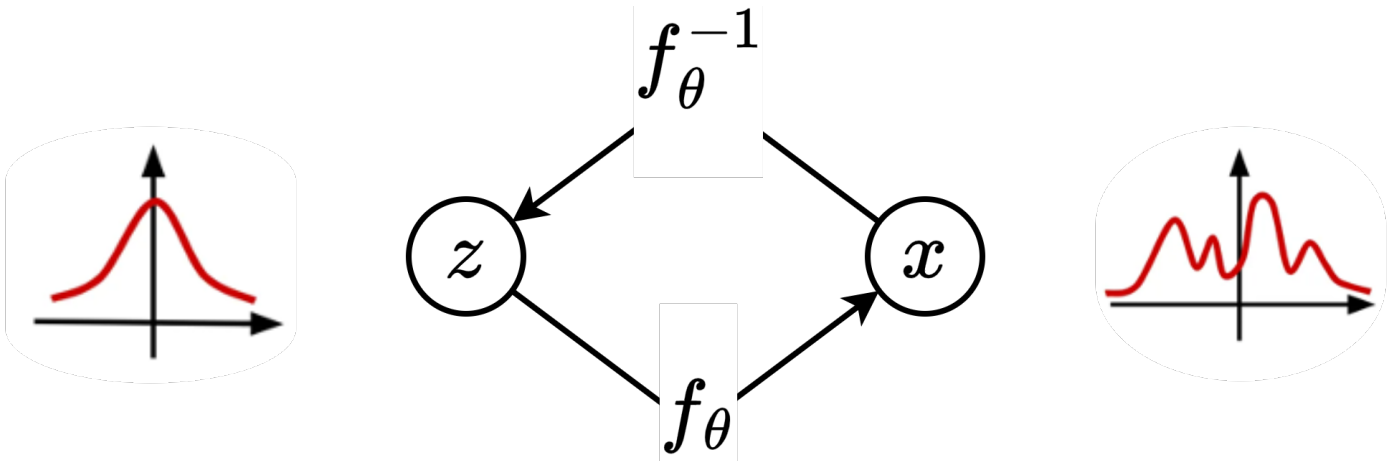


Forward pass of IAF

- Computational tradeoffs
  - MAF: Fast likelihood evaluation, slow sampling
  - IAF: Fast sampling, slow likelihood evaluation
- MAF more suited for training based on MLE, density estimation
- IAF more suited for real-time generation

Summary of Normalizing flow models

- Transform simple distributions into more complex distributions via change of variables
- Jacobian of transformations should have tractable determinant for efficient learning and density estimation
- Computational tradeoffs in evaluating forward and inverse transformations



Thanks