CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

CSDS 600: Deep Generative Models

Normalizing Flow Models (1)

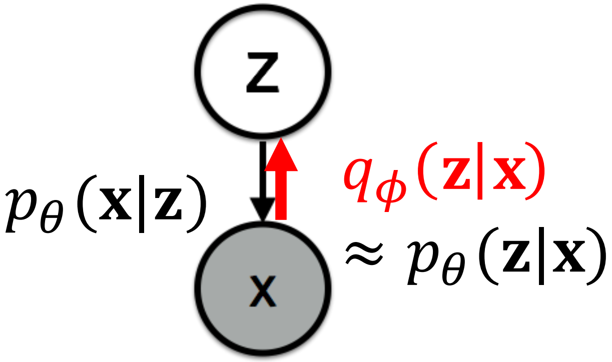
Yu Yin (yu.yin@case.edu)

Case Western Reserve University

https://yin-yu.github.io/

Recap: VAE

- $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})dz$ (Expensive to compute).
- Alternatively, we introduce a variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ to approximates the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$.



- Use KL divergence to quantify the distance of these two posteriors:

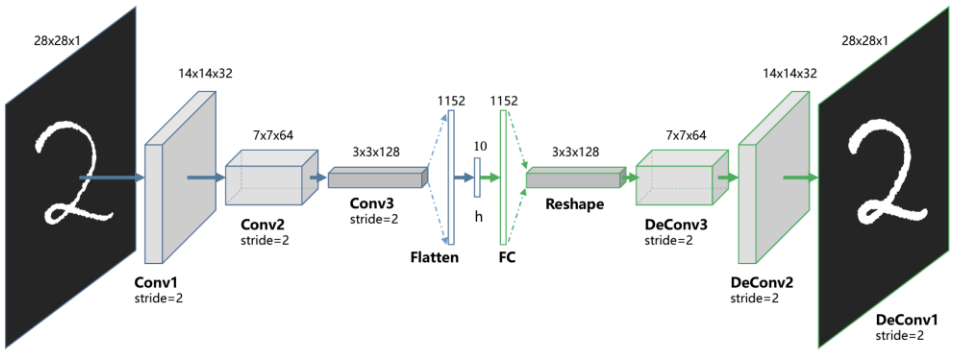
$$D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) = \log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z})$$

$$\Rightarrow \underbrace{\boxed{\log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x}))}}_{\text{Maximize during training}} = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$

- Loss function:

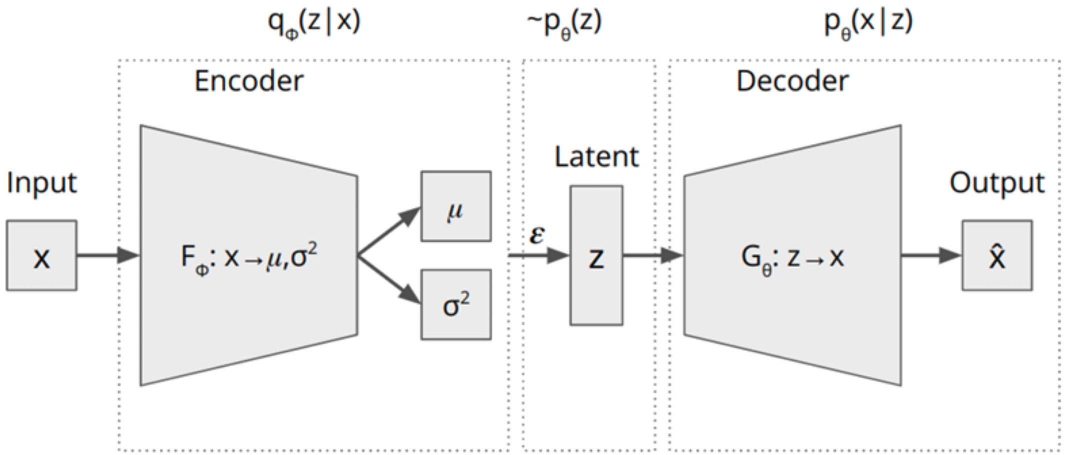
$$\begin{aligned} L_{\mathrm{VAE}}(\theta,\phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \end{aligned}$$

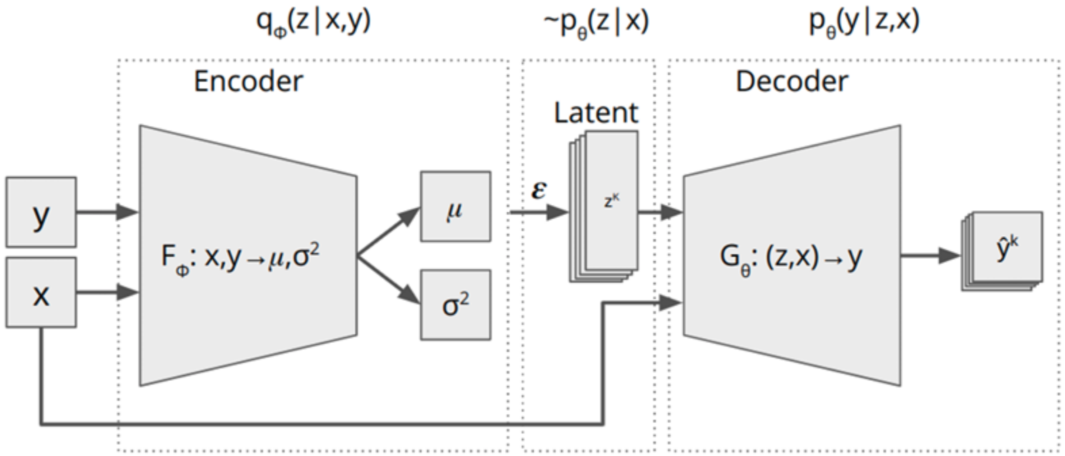
Recap: Convolutional & Conditional VAE



Convolutional VAE

VAE





Conditional VAE

Outline

- Background
- Change of variable theorem
- Flow-based model
- Learning and inference
- Desiderata

Background

- Autoregressive Models:

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i | x_1, x_2, \dots, x_{i-1}) = \prod_{i=1}^{n} p_\theta(x_i | x_{<i})$$

- Variational autoencoders:

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

Background

- Autoregressive Models:

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i | x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p_\theta(x_i | x_{<i})$$

- Variational autoencoders:

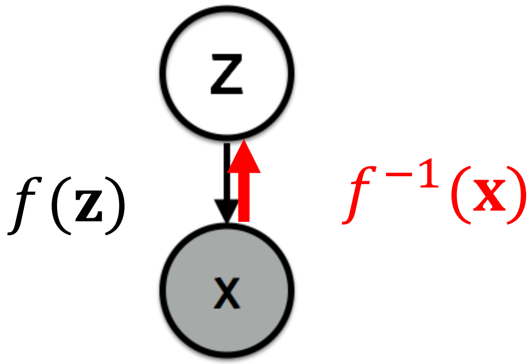
$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

- Can we design a latent variable model with tractable likelihoods?

Normalizing flow models.

Normalizing flow models

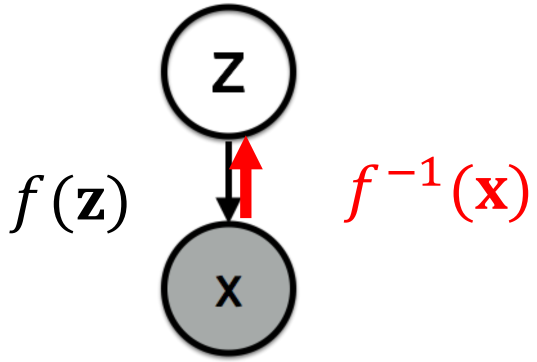
- Directed, latent-variable invertible models



- **Key idea:** The mapping between $\mathbf{z}$ and $\mathbf{x}$, given by $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$, is deterministic and invertible such that $\mathbf{x} = f(\mathbf{z})$ and $\mathbf{z} = f^{-1}(\mathbf{x})$.

Normalizing flow models

- **Key idea:** The mapping between $\mathbf{z}$ and $\mathbf{x}$, given by $f: \mathbb{R}^n \to \mathbb{R}^n$, is deterministic and invertible such that $\mathbf{x} = f(\mathbf{z})$ and $\mathbf{z} = f^{-1}(\mathbf{x})$.

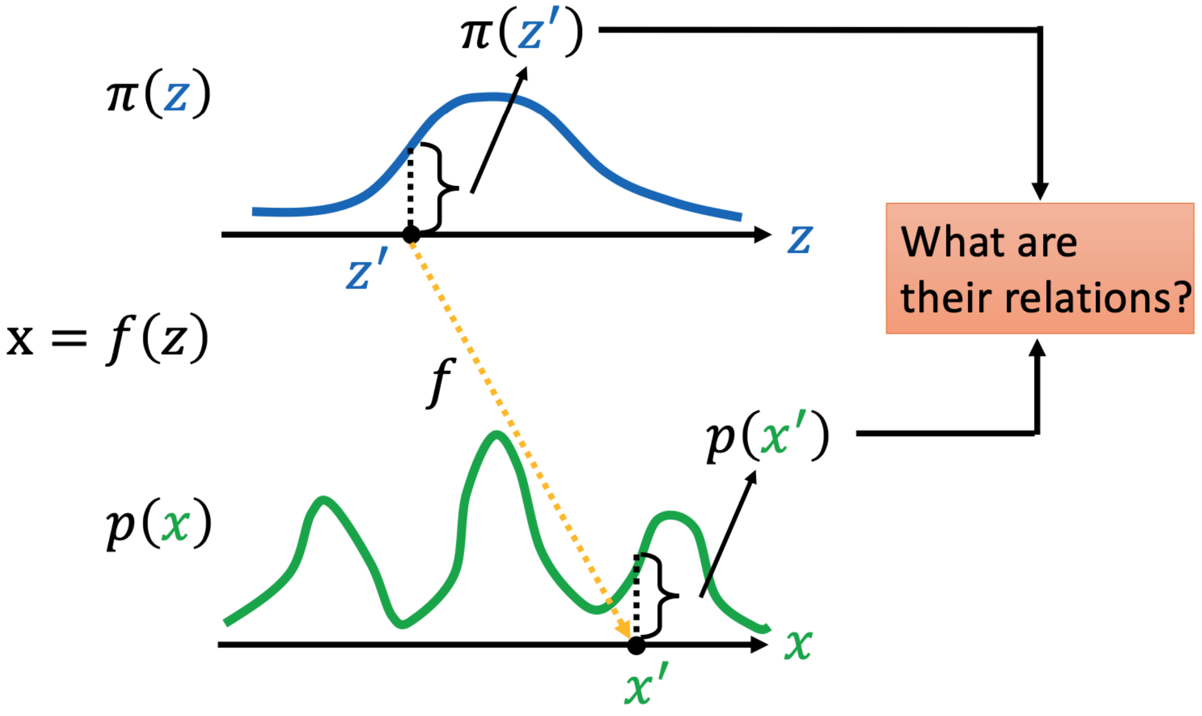


How?

Normalizing flow models

Change of Variable Theorem

- Given a random variable z and its known probability density function $z \sim \pi(z)$
- Invertible mapping function such that $x = f(z), z = f^{-1}(x)$

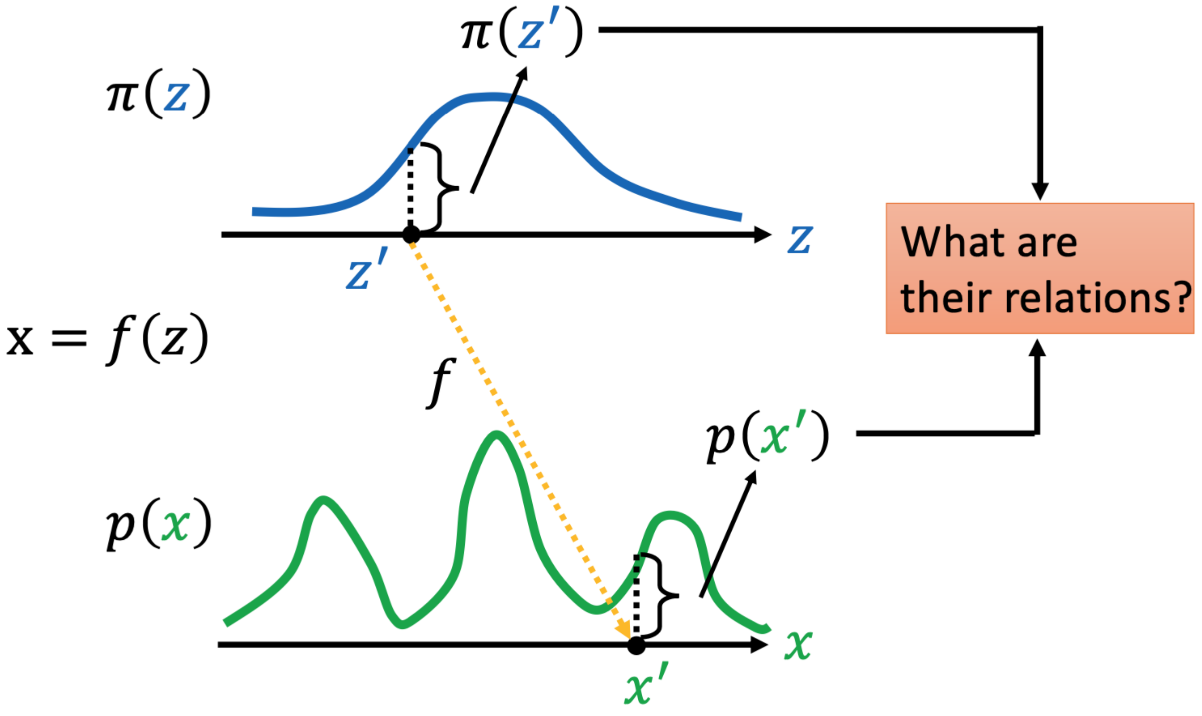


$$\int p(x)\,dx = \int \pi(z)\,dz = 1$$

Normalizing flow models

Change of Variable Theorem

- Given a random variable z and its known probability density function $z \sim \pi(z)$
- Invertible mapping function such that $x = f(z), z = f^{-1}(x)$
- How to infer the unknown probability density function $p(x)$?



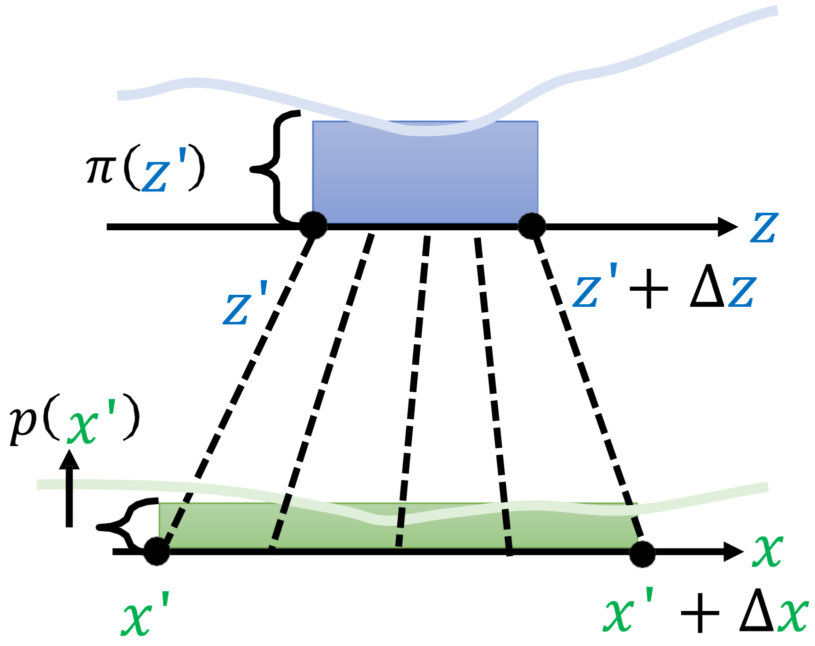
$$\int p(x)\,dx = \int \pi(z)\,dz = 1$$

Normalizing flow models

Change of Variable Theorem

- How to infer the unknown probability density function $p(x)$?

Since the function f is **invertible** and **differentiable**, we have



The blue square and the green square should be equal in area

$$|p(x')\Delta x| = |\pi(z')\Delta z|$$

Normalizing flow models

Change of Variable Theorem

- How to infer the unknown probability density function $p(x)$?

  Since the function f is **invertible** and **differentiable**, we have

$$|p(x)dx| = |\pi(z)dz|$$

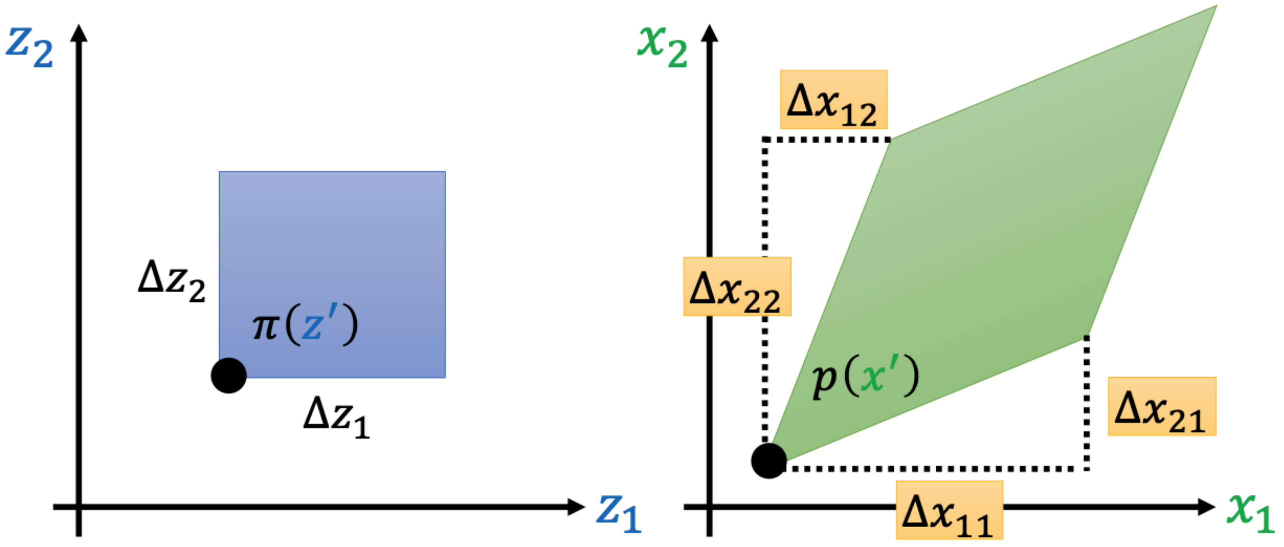
$$=> \quad p(x) = \pi(z)\left|\frac{dz}{dx}\right|$$

$$= \pi\left(f^{-1}(x)\right)\left|\frac{df^{-1}}{dx}\right|$$

Normalizing flow models

Change of Variable Theorem (**multivariable**)

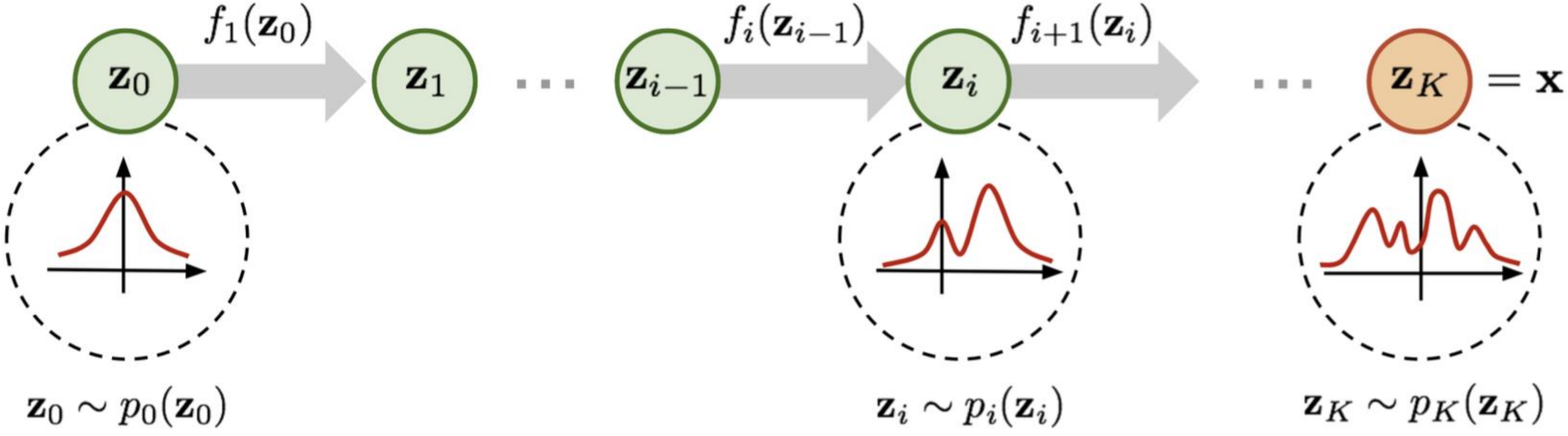
- $\mathbf{z} \sim \pi(\mathbf{z})$
- $\mathbf{x} = f(\mathbf{z}), \mathbf{z} = f^{-1}(\mathbf{x})$
- How to infer $p(\mathbf{x})$?



$$p(\mathbf{x}) = \pi(\mathbf{z}) \left| \det \frac{dz}{dx} \right| = \pi\left(f^{-1}(\mathbf{x})\right) \left| \det \frac{df^{-1}}{d\mathbf{x}} \right|$$

Normalizing flow models

- Transform a simple distribution to a complex one step by step.



$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1})$$

$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{ thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|$$

Normalizing flow models

- Convert the equation to be a function of $\mathbf{z}_i$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{df_i^{-1}}{d\mathbf{z}_i} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left(\frac{df_i}{d\mathbf{z}_{i-1}} \right)^{-1} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|^{-1}$$

If $y = f(x)$ and $x = f^{-1}(y)$, we have:

$$\frac{df^{-1}(y)}{dy} = \frac{dx}{dy} = \left(\frac{dy}{dx}\right)^{-1} = \left(\frac{df(x)}{dx}\right)^{-1}$$

$$\det(M^{-1}) = (\det(M))^{-1}$$

Normalizing flow models

- Convert the equation to be a function of $\mathbf{z}_i$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{d f_i^{-1}}{d \mathbf{z}_i} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left(\frac{d f_i}{d \mathbf{z}_{i-1}} \right)^{-1} \right|$$

$$= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \frac{d f_i}{d \mathbf{z}_{i-1}} \right|^{-1}$$

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{d f_i}{d \mathbf{z}_{i-1}} \right|$$

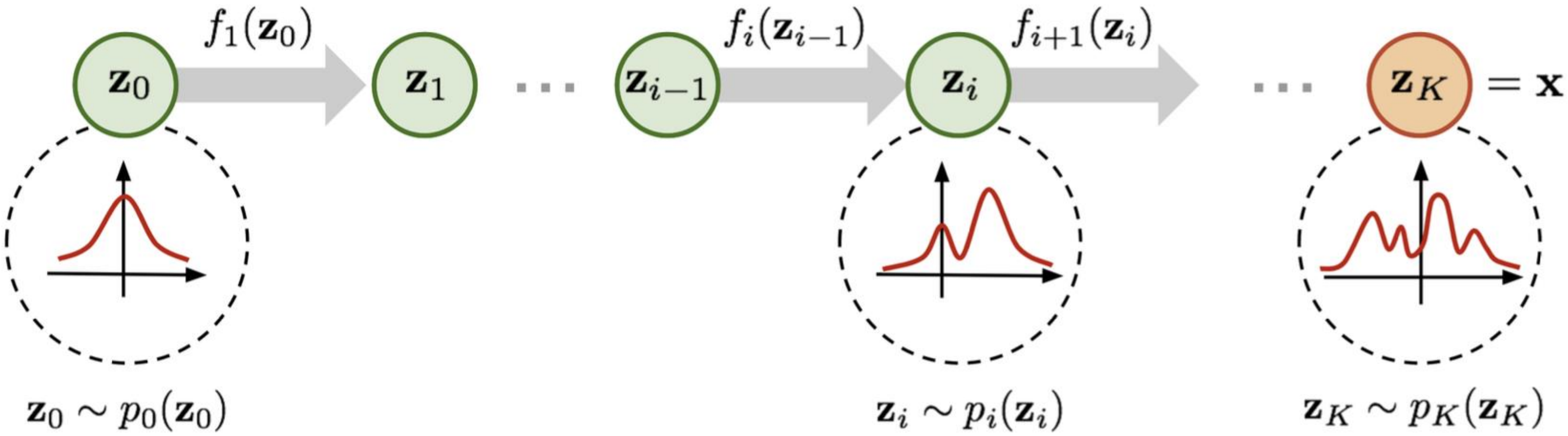
If $y = f(x)$ and $x = f^{-1}(y)$, we have:

$$\frac{d f^{-1}(y)}{dy} = \frac{dx}{dy} = (\frac{dy}{dx})^{-1} = (\frac{d f(x)}{dx})^{-1}$$

$$\det(M^{-1}) = (\det(M))^{-1}$$

Normalizing flow models

- Trace back to the initial distribution $\mathbf{z_0}$



$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)$$

Normalizing flow models

- Trace back to the initial distribution $\mathbf{z_0}$

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|$$

$$\begin{aligned}
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{df_{K-1}}{d\mathbf{z}_{K-2}} \right| - \log \left| \det \frac{df_K}{d\mathbf{z}_{K-1}} \right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{df_i}{d\mathbf{z}_{i-1}} \right|
\end{aligned}$$

Normalizing flow models

Learning and inference

- Learning via **maximum likelihood** over the dataset

1） **Exact likelihood evaluation** via inverse transformation and change of variables formula

2） **Sampling** via forward transformation $f : \mathbf{z} \to \mathbf{x}$

$$\mathbf{x} = \mathbf{z}_K = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0)$$

3） **Latent representations** inferred via inverse transformation (no inference network required!)

Normalizing flow models

- “Normalizing” means that the change of variables gives a normalized density after applying an invertible transformation.

- “Flow” means that the invertible transformations can be composed with each other to create more complex invertible transformations.

Normalizing flow models

Desiderata

- Simple prior $\pi(z)$ that allows for efficient sampling and tractable likelihood evaluation. E.g., Gaussian
- Invertible transformations
- Computing likelihoods also requires the evaluation of determinants of $n \times n$ Jacobian matrices, where n is the data dimensionality
  - Computing the determinant for an $n \times n$ matrix is $O(n$^3): prohibitively expensive within a learning loop!
  - 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 $O(n)$ operation

Summary

- Transform simple to complex distributions via sequence of invertible transformations
- Learning via maximum likelihood over the dataset
- What we need?
  - Prior $\pi(z)$ easy to sample
  - Invertible transformations
  - Determinants of Jacobian Efficient to compute

Project proposal

- Project proposal: **Oct. 4**
- Proposal presentation counts for 10% of the overall project grade
- Can be done in groups of up to 2 students
- Can fall into one or more of the following categories:
  - Application of deep generative models on a novel task/dataset
  - Algorithmic improvements into the evaluation, learning and/or inference of deep generative models
  - Theoretical analysis of any aspect of existing deep generative models
  - Reproduction of empirical results reported in a recent paper

Project proposal

- Prepare 3-4 slides for proposal
  - Project title, team members
  - What is the problem and motivation
  - Possible solutions or plans
- 3-5 minutes for presentation

Thank You

- Questions?

- Email: yu.yin@case.edu

- TA: Jiayi Chen <jxc2077@case.edu>

Reference slides

- https://lilianweng.github.io/posts/2018-10-13-flow-models/
- Hao Dong. Deep Generative Models
- Hung-yi Li. Flow-based Generative Model