Introduction to Deep Generative Modeling

Spring School on Models and Data, University of South Carolina

Lars Ruthotto
lruthotto@emory.edu

Departments of Mathematics and Computer Science
Emory University

Motivation: Deep Generative Modeling

Goal: Given samples $\mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathbb{R}^n$ learn a representation of their underlying distribution $\mathcal{X}$.

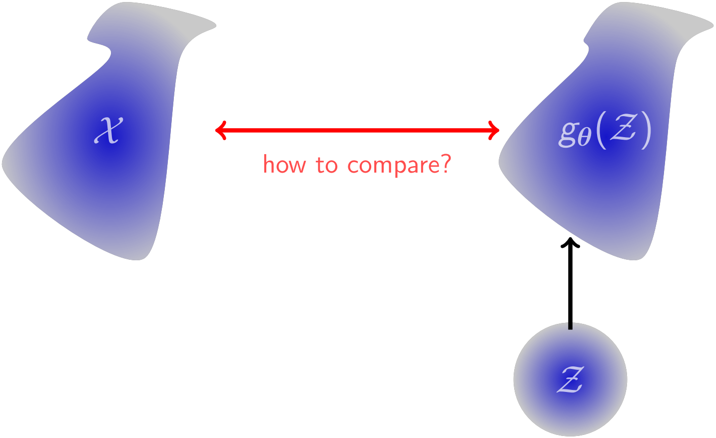
Challenges: n typically large, $\mathcal{X}$ complicated (multimodal, disjoint support, etc.)

Idea: Train parameters $\boldsymbol{\theta}$ of *generator* $g_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}^n$ so that it transforms a given *latent distribution* $\mathcal{Z} \subset \mathbb{R}^q$ to match $\mathcal{X}$.

Generator can be used for

- density estimation: $p_{\mathcal{X}}(\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z}) p_{\mathcal{Z}}(\mathbf{z}) d\mathbf{x}$
- sampling: $g_{\boldsymbol{\theta}}(\mathbf{z})$ where $\mathbf{z} \sim \mathcal{Z}$ (main focus today)

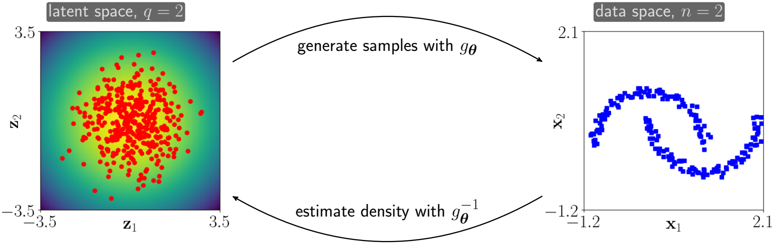
Illustration: Deep Generative Modeling



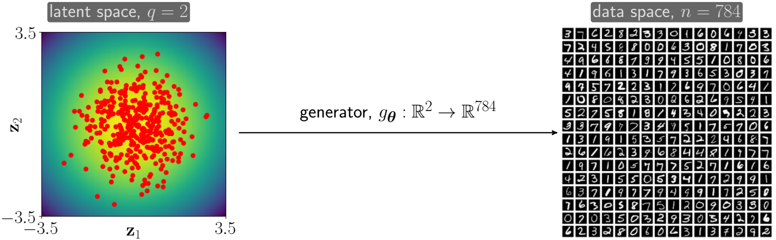
- $\mathcal{X}$ - data distribution
- $g_{\boldsymbol{\theta}}$ - generator (today: deep neural network)
- $\boldsymbol{\theta}$ - parameters/weights
- $\mathcal{Z}$ - latent distribution (today: $\mathcal{N}(0, \mathbf{I}_q)$)

Examples: Moons and MNIST Dataset

Moons toy example:



MNIST image generation example:



Workshop Overview: Intro to Deep Generative Modeling

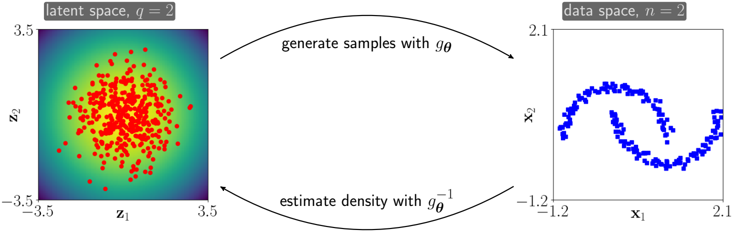
Objective: Discuss the three most popular classes of approaches in a common mathematical framework (main ref [13]).

1. (Continuous) Normalizing Flows (NF / CNF)
   - construct $g_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^n$ to be diffeomorphic
   - train $g_{\boldsymbol{\theta}}$ by maximizing likelihood of samples
2. Variational Autoencoders (VAE)
   - support non-invertible, non-smooth $g_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}^n$
   - replace inverse of generator by approx. posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$
   - train generator using lower bound of samples' likelihood
3. Generative Adversarial Networks (GAN)
   - support non-invertible, non-smooth $g_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}^n$
   - likelihood-free training using classifier or transport-distance

each part: 20 min lecture + time for coding, discussion, break.

(Continuous) Normalizing Flows

(Continuous) Normalizing Flows (CNF)



Assumption: $g_{\boldsymbol{\theta}}$ is diffeomorphism (requires $q = n$)

Use change of variables formula to approximate likelihood

$$
\begin{aligned}
p_{\mathcal{X}}(\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{x}) &= p\left(g_{\boldsymbol{\theta}}^{-1}(\mathbf{x})\right) \cdot \det \nabla g_{\boldsymbol{\theta}}^{-1}(\mathbf{x}) \\
&= (2\pi)^{-\frac{n}{2}} \exp\left(-\frac{1}{2}\|g_{\boldsymbol{\theta}}^{-1}(\mathbf{x})\|^2\right) \cdot \det \nabla g_{\boldsymbol{\theta}}^{-1}(\mathbf{x})
\end{aligned}
$$

Maximum Likelihood Training

$$J_{\mathrm{ML}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x})\right]$$
$$\approx \frac{1}{s}\sum_{i=1}^{s}\left(\frac{1}{2}\left\|g_{\boldsymbol{\theta}}^{-1}\left(\mathbf{x}^{(i)}\right)\right\|^2 - \log\det\nabla g_{\boldsymbol{\theta}}^{-1}\left(\mathbf{x}^{(i)}\right) + c\right)$$

with i.i.d. samples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(s)} \sim \mathcal{X}$.

Remark: $\min_{\boldsymbol{\theta}} J_{\mathrm{ML}}(\boldsymbol{\theta})$ is equivalent to minimizing the Kullback-Leibler divergence between $p_{\mathcal{X}}$ and $p_{\boldsymbol{\theta}}$

$$\mathrm{KL}(p_{\mathcal{X}}||p_{\boldsymbol{\theta}}) = \int p_{\mathcal{X}}(\mathbf{x})\log\frac{p_{\mathcal{X}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}d\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[\log\left(\frac{p_{\mathcal{X}}(\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\right)\right]$$

since $p_{\mathcal{X}}(\mathbf{x})$ does not depend on $\boldsymbol{\theta}$.

Note: Training needs $g_{\boldsymbol{\theta}}^{-1}$, generation needs $g_{\boldsymbol{\theta}}$.

Finite Normalizing Flows

Idea: For a fixed $\mathbf{x} \sim \mathcal{Z}$, write generator as

$$g_{\boldsymbol{\theta}}(\mathbf{z}) = f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z})$$

and $\mathbf{y}^{(K)}, \mathbf{y}^{(K-1)}, \ldots, \mathbf{y}^{(1)}$ be the hidden features.

Then the inverse and log-determinant of the flow are

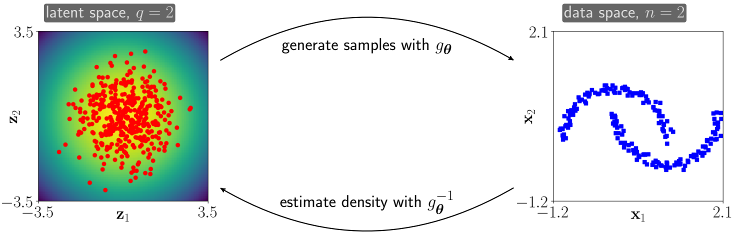
$$g_{\boldsymbol{\theta}}^{-1}(\mathbf{x}) = f_1^{-1} \circ f_2^{-1} \circ \cdots \circ f_K^{-1}(\mathbf{x}),$$

$$\log\det \nabla g_{\boldsymbol{\theta}}^{-1}(\mathbf{x}) = \sum_{j=K}^{1} \log\det \nabla f_j^{-1}\left(\mathbf{y}^{(j)}\right).$$

Trade off when choosing layer functions f_j:

- expressiveness: able to approximate complicated transformation
- tractability: easy-to-evaluate inverse and log-determinant

Normalizing Flows: Some References



- Efficient $g_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\theta}}^{-1}$
  - NICE: Non-linear independent components estimation [4]
  - real NVP: real non-volume preserving flows [5] (next slide)
- Efficient $g_{\boldsymbol{\theta}}$ but not $g_{\boldsymbol{\theta}}^{-1}$
  - planar and radial flows [12]
  - inverse autoregressive flows [9]
- Not efficient $g_{\boldsymbol{\theta}}$ but efficient $g_{\boldsymbol{\theta}}^{-1}$
  - masked auto-regressive flow [11]

Example: Real Non-Volume Preserving Flow [5]

Let $n = q = 2$. The jth layer splits its input $\mathbf{y}^{(j)} \in \mathbb{R}^2$ into its components $\mathbf{y}_1^{(j)}$ and $\mathbf{y}_2^{(j)}$

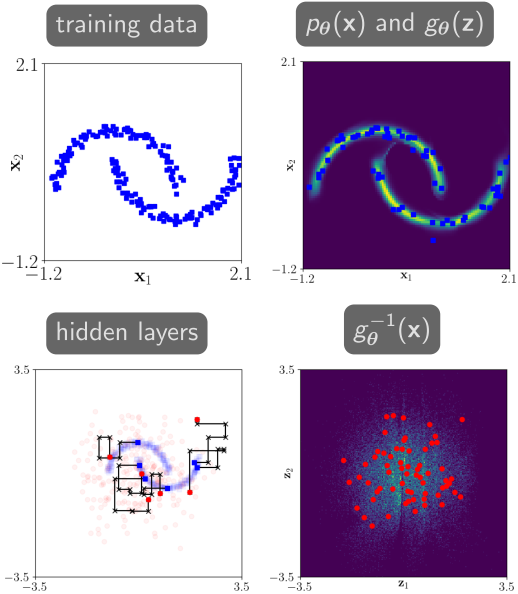
$$f_j\left(\mathbf{y}^{(j)}\right) = \begin{bmatrix} \mathbf{y}_1^{(j)} \\ \mathbf{y}_2^{(j)} \cdot \exp\left(s_j\left(\mathbf{y}_1^{(j)}\right)\right) + t_j\left(\mathbf{y}_1^{(j)}\right) \end{bmatrix},$$

where $s_j, t_j : \mathbb{R} \to \mathbb{R}$ are neural networks that model scaling and translation, respectively.

Checklist:

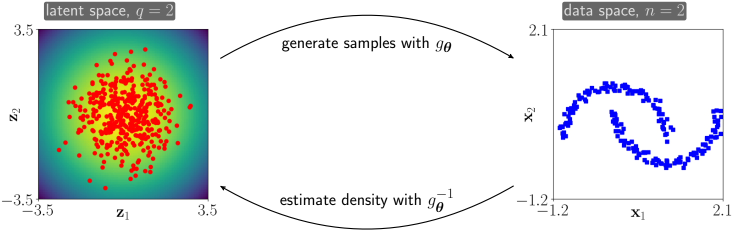
- log-determinant and inverse trivial to compute (try it!)
- expressiveness may require many layers (think n large)

Example: Real NVP for Moons Dataset



see `RealNVP.ipynb`

Continuous Normalizing Flows [8]

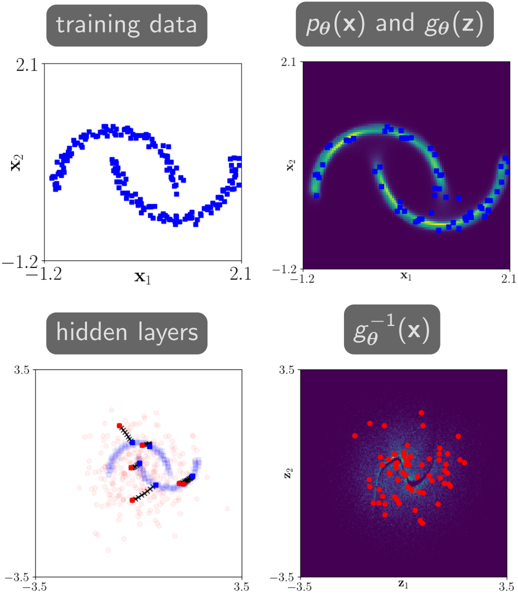


For $T > 0$, let $g_{\boldsymbol{\theta}}(\mathbf{z}) = \mathbf{y}(T)$ where $\mathbf{y} : [0, T] \to \mathbb{R}^n$ satisfies

$$\mathbf{y}'(t) = v_{\boldsymbol{\theta}}(\mathbf{y}(t), t), \quad \text{where} \quad \mathbf{y}(0) = \mathbf{z}.$$

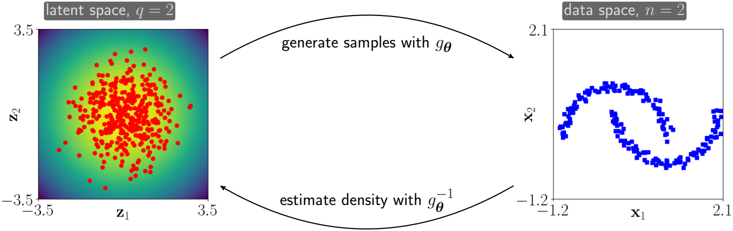
Here, $v_{\boldsymbol{\theta}} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is a neural network $\rightsquigarrow$ Neural ODE [3]

Example: OT-Flow for Moons Dataset



see `OTFlow.ipynb`

Discussion: Normalizing Flows



- train $g_{\boldsymbol{\theta}}$ by maximizing likelihood of samples
- can be used for sampling and density estimation
- limitation: $g_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^n$ is diffeomorphic $\Rightarrow$ *intrinsic* dimension of $\mathcal{X}$ must be n and support cannot be disjoint
- NF: need to trade-off expressiveness and efficiency
- CNF: scalable to high dimensions, that is, $n = \mathcal{O}(10^2)$.

Variational Autoencoders

Variational Autoencoders (VAE) [10]

Let now $q \neq n$, for example, $q \ll n$ (very common).

Cannot use (C)NF since

- $g_{\boldsymbol{\theta}}^{-1}$ may not exist
- $\mathrm{KL}(p_{\mathcal{X}}(\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{x}))$ may be unbounded

Need alternative way to define a loss function!

Apply Bayes' rule and note that

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\mathcal{Z}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}, \quad \text{for} \quad \mathbf{z} \sim \mathcal{Z}.$$

Cannot maximize right hand side directly ($p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is intractable)

Key idea in VAE: Approximate the Posterior

For an arbitrary generator $g_{\boldsymbol{\theta}}$, it is intractable to compute the posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

Idea: Learn an approximate posterior

$$e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$$

Today we consider

$$e_{\psi}(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{\psi}}(\mathbf{x}), \exp(\boldsymbol{\Sigma}_{\boldsymbol{\psi}}(\mathbf{x}))\right).$$

where

- $\boldsymbol{\psi}$ are the weights (in general $\boldsymbol{\psi} \neq \boldsymbol{\theta}$)
- $\boldsymbol{\Sigma}_{\psi}(\mathbf{x})$ is diagonal
- approximate posterior is similar to an encoder

Evidence Lower Bound Training

Idea: Replace $\min_{\boldsymbol{\theta}} -\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)$ by $\min_{\boldsymbol{\psi},\boldsymbol{\theta}} \log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\right)$

Why would this be meaningful?

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] \\
&= \mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\cdot\frac{e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\right)+\log\left(\frac{e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right)\right]
\end{aligned}
$$

Drop second term (i.e., $\mathrm{KL}\left(e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\right) \geq 0$ to obtain lower bound (called empirical lower bound or ELBO).

VAE Training Problem

$$\begin{aligned} J_{\mathrm{VAE}}(\boldsymbol{\psi}, \boldsymbol{\theta}) &= -\mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \log p_{\mathcal{Z}}(\mathbf{z}) + \log e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})\right] \\ &\approx \frac{1}{s}\sum_{i=1}^{s}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}) - \log p_{\mathcal{Z}}(\mathbf{z}^{(i)}) + \log e_{\boldsymbol{\psi}}(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})\right] \end{aligned}$$

with

- i.i.d. samples $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(s)}$ from $\mathcal{X}$
- one sample $\mathbf{z}^{(i)}$ per approximate posterior $e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x}^{(i)})$ (you can use more, of course)

Remarks:

- above estimate of objective is unbiased, but can be noisy
- $\min_{\boldsymbol{\psi}} J_{\mathrm{VAE}}(\boldsymbol{\psi}, \bar{\boldsymbol{\theta}})$ for given $\bar{\boldsymbol{\theta}}$ improves tightness of ELBO.

Interpret VAE as Regularized Autoencoder

Note that we can re-write the objective in VAE as

$$
\begin{aligned}
J_{\mathrm{ELBO}}(\boldsymbol{\psi}, \boldsymbol{\theta}) &= \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\left[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \log e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x}) - \log p_{\mathcal{Z}}(\mathbf{z}\right. \\
&= \mathbb{E}_{\mathbf{x}\sim\mathcal{X}}\left[-\mathbb{E}_{\mathbf{z}\sim e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})}\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + \mathrm{KL}\left(e_{\boldsymbol{\psi}}(\mathbf{z}|\mathbf{x})||p_{\mathcal{Z}}(\mathbf{z})\right)\right]
\end{aligned}
$$

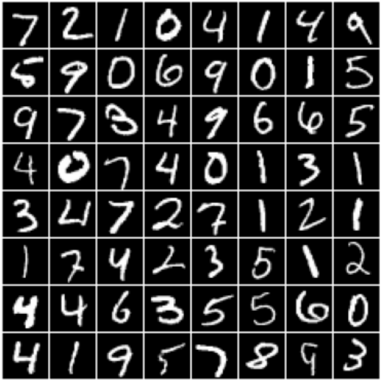
- first term: minimize approximation error
- second term: bias approximate posteriors toward $\mathcal{Z}$
- need to carefully balance both terms (Bayesian vs. frequentist)

Example: Autoencoders are trained with no regularization

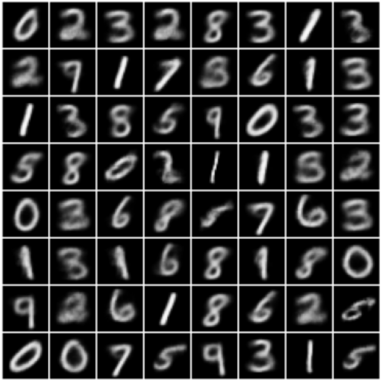
- minimize approximation error
- latent space can be irregular and different from $\mathcal{Z}$
- cannot expect $g_{\boldsymbol{\theta}}(\mathbf{z})$ to be similar to $\mathcal{X}$ when $\mathbf{z} \sim \mathcal{Z}$

Example: VAE for MNIST

test images

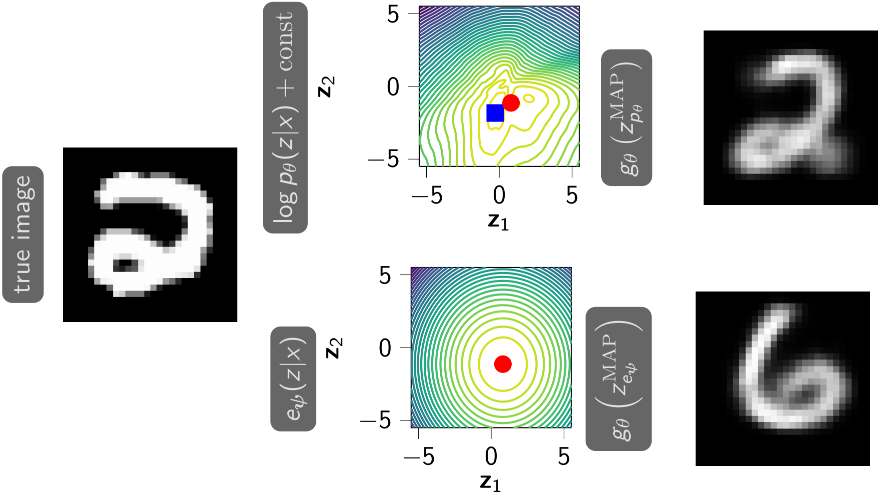


generated images



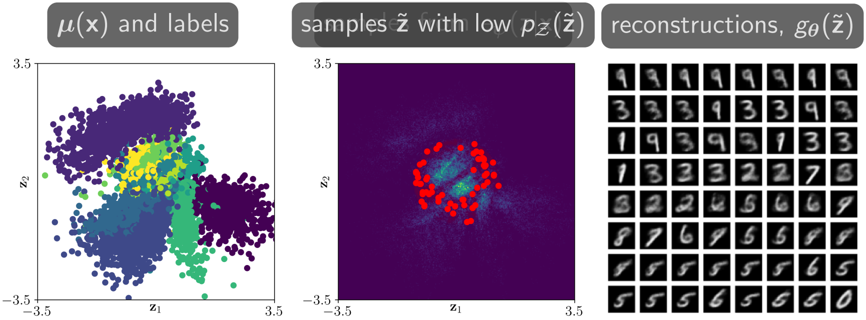
see `VAE.ipynb`

Example: Quality of Posterior Approximation MNIST



see `VAE.ipynb`

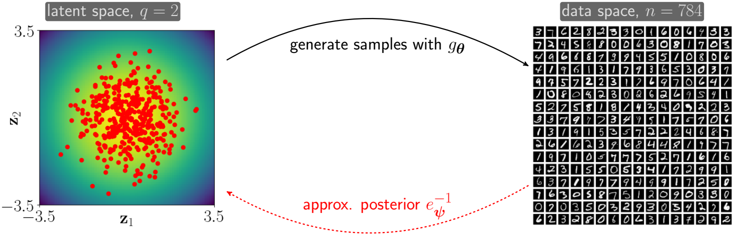
Example: Structure of Latent Space



- $g_{\boldsymbol{\theta}}^{-1}(\mathcal{X}) \neq \mathcal{Z} \Rightarrow$ generator not trained using samples from $\mathcal{Z}$
- in general, expect poor performance of generator for $\tilde{\mathbf{z}}$

see `VAE.ipynb`

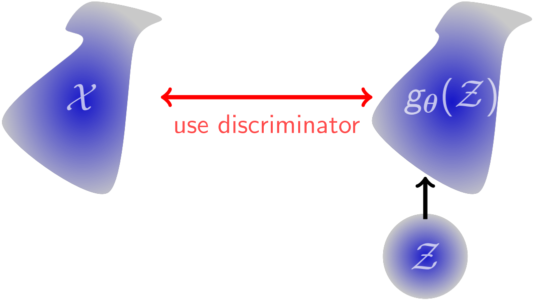
Σ : Variational Auto Encoder



- support non-invertible, non-smooth $g_{\boldsymbol{\theta}} : \mathbb{R}^q \to \mathbb{R}^n$
- use true posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ by $e_{\psi}(\mathbf{z}|\mathbf{x})$
- train $g_{\boldsymbol{\theta}}$ and $e_{\psi}(\mathbf{z}|\mathbf{x})$ using lower bound of samples' likelihood
- interpret as regularized autoencoder to get more flexibility
- $g_{\boldsymbol{\theta}}$ trained using samples from approx. posteriors ($\neq \mathcal{Z}$).

Generative Adversarial Networks

Generative Adversarial Networks (GAN) [7, 6, 1]



Idea: Train by minimizing the distance between $\mathcal{X}$ and $g_{\boldsymbol{\theta}}(\mathcal{Z})$.

Some properties of GAN training

- likelihood free: no density estimate / lower bound needed
- avoids the correspondence problem
- sample from latent distribution in training (unlike CNF, VAE)

Key component: (trained) discriminator to measure the distance.
Today: Binary classification / Transport costs

Discriminator based on Binary Classification [7]

Idea: Consider two sample test problem. Find a discriminator

$$d_{\phi} : \mathbb{R}^n \to [0,1] \quad \text{such that} \quad d_{\phi}(\mathbf{x}) \approx \begin{cases} 1, & \mathbf{x} \sim \mathcal{X} \\ 0, & \mathbf{x} \sim g_{\boldsymbol{\theta}}(\mathcal{Z}). \end{cases}$$

Note: d_{ϕ} will be a DNN with weights ϕ.

GAN training seeks to find a Nash equilibrium of

$$J_{\mathrm{GAN}}(\boldsymbol{\theta}, \phi) = \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[\log(d_{\phi}(\mathbf{x})) \right] + \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} \left[\log \left(1 - d_{\phi}(g_{\boldsymbol{\theta}}(\mathbf{z})) \right) \right].$$

In other words, find $(\boldsymbol{\theta}^*, \phi^*)$ such that

$$\phi^* \in \arg\max_{\phi} J_{\mathrm{GAN}}(\boldsymbol{\theta}^*, \phi) \quad \text{and} \quad \boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} J_{\mathrm{GAN}}(\boldsymbol{\theta}, \phi^*).$$

In practice: Use stochastic approximation and alternate between updating ϕ and $\boldsymbol{\theta}$ (need to balance learning rates, batch sizes, ...)

Reminder: Solving Saddle-Point Problems ain't easy!

Let $\boldsymbol{\theta}^*$ be the weights of an optimal generator with $g_{\boldsymbol{\theta}^*}(\mathcal{Z}) = \mathcal{X}$.

What would that mean for the optimal discriminator $d_{\boldsymbol{\phi}^*}$?

Remarks:

1. $g_{\boldsymbol{\theta}^*}$ and $d_{\boldsymbol{\phi}^*}$ are parameterized by DNN
2. need expressiveness and ideal weights to find this equilibrium

$\Rightarrow$ GAN effectiveness is very hard to predict

This equilibrium will not be stable! Let $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^* + \delta\boldsymbol{\theta}$ with $\|\delta\boldsymbol{\theta}\|$ small and $g_{\tilde{\boldsymbol{\theta}}}(\mathcal{Z}) \neq \mathcal{X}$. Then, the optimal discriminator would be able to distinguish between samples and data points.

Even worse: We could have $\nabla_{\boldsymbol{\theta}} J_{\mathrm{GAN}}(\tilde{\boldsymbol{\theta}}, \phi^*) \approx 0$.

For more detailed theory and other issues, see [1]

Mode Collapse in GAN Training

Example: $g_{\boldsymbol{\theta}}$ maps almost all $\mathbf{z} \sim \mathcal{Z}$ to first data point, that is,

$$g_{\boldsymbol{\theta}}(\mathbf{z}) = \mathbf{x}^{(1)} \quad \text{for almost all} \quad \mathbf{z} \sim \mathcal{Z}$$

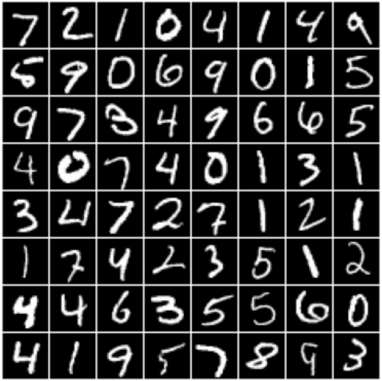
What would that mean for the optimal discriminator d_{ϕ^*}?

In this example, mode collapse is easy to detect! What would happen if $g_{\boldsymbol{\theta}}$ mapped almost no $\mathbf{z} \sim \mathcal{Z}$ close to $\mathbf{x}^{(1)}$?

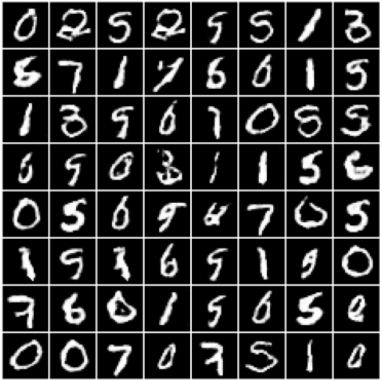
Mode collapse is difficult to detect/avoid. For some heuristics (batch statistics, label smoothing, . . .) see [14].

Example: DCGAN for MNIST

test images



generated images, VAE init



see `DCGAN.ipynb`

Wasserstein GAN: Transport Costs as Discriminator [2]

Idea: Train $g_{\boldsymbol{\theta}}$ to minimize Wasserstein-1 distance between $\mathcal{X}$ and $g_{\boldsymbol{\theta}}(\mathcal{Z})$.

$$W_1(g_{\boldsymbol{\theta}}(\mathcal{Z}), \mathcal{X}) = \inf_{\gamma \in \Pi} \mathbb{E}_{(\widehat{\mathbf{x}}, \mathbf{x}) \sim \gamma} \left[\| \widehat{\mathbf{x}} - \mathbf{x} \| \right]$$

Here:

- γ is a (probabilistic) transport map
- $\gamma(\widehat{\mathbf{x}}, \mathbf{x})$: probability of moving mass between $\widehat{\mathbf{x}}$ and $\mathbf{x}$
- Π: set of all $\gamma(\cdot, \cdot)$ with marginals $\mathcal{X}$ and $g_{\boldsymbol{\theta}}(\mathcal{Z})$, respectively.

Most practical implementations use equivalent definition

$$W_1(g_{\boldsymbol{\theta}}(\mathcal{Z}), \mathcal{X}) = \max_{f \in \mathrm{Lip}(f) \leq 1} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} \left[f\left(g_{\boldsymbol{\theta}}(\mathbf{z}) \right) \right] - \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[f(\mathbf{x}) \right].$$

Crux: Need to design and train another NN model $f_{\phi} : \mathbb{R}^n \to \mathbb{R}$

Properties of GAN Training [2]

$$\min_{\boldsymbol{\theta}} \max_{f \in \mathrm{Lip}(f) \leq 1} \mathbb{E}_{\mathbf{z} \sim \mathcal{Z}} \left[f\left(g_{\boldsymbol{\theta}}(\mathbf{z}) \right) \right] - \mathbb{E}_{\mathbf{x} \sim \mathcal{X}} \left[f(\mathbf{x}) \right]$$

Theoretical advantages over discriminator-based GAN

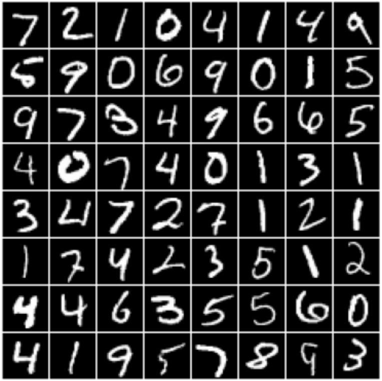
- $g_{\boldsymbol{\theta}}$ continuous $\Rightarrow (\boldsymbol{\theta}) \mapsto W_1(g_{\boldsymbol{\theta}}(\mathcal{Z}), \mathcal{X})$ continuous
- $g_{\boldsymbol{\theta}}$ loc. Lipschitz $\Rightarrow (\boldsymbol{\theta}) \mapsto W_1(g_{\boldsymbol{\theta}}(\mathcal{Z}), \mathcal{X})$ differentiable

Practical considerations

- Need to enforce $f \in \mathrm{Lip}(f) \leq 1$ (crop weights, gradient penalty, ...)
- Training f more accurately may not improve results [15]

Example: WGAN for MNIST

test images



generated images, VAE init

see `WGAN.ipynb`

Summary

Workshop Overview: Intro to Deep Generative Modeling

Objective: Discuss the three most popular classes of approaches in a common mathematical framework (main ref [13]).

1. (Continuous) Normalizing Flows (NF / CNF)
   - construct $g_{\boldsymbol{\theta}} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ to be diffeomorphic
   - train $g_{\boldsymbol{\theta}}$ by maximizing likelihood of samples
2. Variational Autoencoders (VAE)
   - support non-invertible, non-smooth $g_{\boldsymbol{\theta}} : \mathbb{R}^q \rightarrow \mathbb{R}^n$
   - replace inverse of generator by approx. posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$
   - train generator using lower bound of samples' likelihood
3. Generative Adversarial Networks (GAN)
   - support non-invertible, non-smooth $g_{\boldsymbol{\theta}} : \mathbb{R}^q \rightarrow \mathbb{R}^n$
   - likelihood-free training using classifier or transport-distance

Comparison of Approaches

- (Continuous) Normalizing Flows
  - \+ compute and optimize likelihood
  - \+ minimize distance between $g_{\boldsymbol{\theta}}^{-1}(\mathcal{X})$ and $\mathcal{Z}$
  - \- assume smoothness of generator
  - \- in most cases q unknown or known that $q < d$
- Variational Autoencoders
  - \+ $g_{\boldsymbol{\theta}}$ can be non-invertible, non-smooth and $q \neq d$
  - \+ loss is related to likelihood, no saddle point problem
  - \- computing likelihood is intractable
  - \- not clear that latent space is sampled well during training
- Generative Adversarial Networks
  - \+ $g_{\boldsymbol{\theta}}$ can be non-invertible, non-smooth and $q \neq d$
  - \+ optimize quality of samples $\rightsquigarrow$ often performs best
  - \- danger of mode collapse (not for WGAN)
  - \- need to compare high-dimensional and complex distributions
  - \- difficult saddle point problems (hyperparameters,...)

Σ: Deep Generative Modeling

DGM likely to remain an active research topic

Some mathematical challenges:

- how to compare high-dimensional, complicated distributions? core problem in statistics for decades
- DGM is ill-posed $\rightsquigarrow$ need to better understand role of hyperparameters (NN design, objective function, regularization, optimization,..)
- no real guidelines for choosing the latent distribution (or even determine the intrinsic dimensionality of the data)
- improve efficiency of training algorithms

Thanks to the organizers and all participants!

Questions/suggestions/remarks? $\rightarrow$ `lruthotto@emory.edu`

References

[1] M. Arjovsky and L. Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *arXiv:1701.04862*, Jan. 2017.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv:1701.07875*, Jan. 2017.

[3] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. In *NeurIPS*, June 2018.

[4] L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear Independent Components Estimation. *arXiv:1410.8516*, Oct. 2014.

[5] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *arXiv:1605.08803*, May 2016.

[6] I. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *arXiv:1701.00160*, Dec. 2016.

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. volume 27, pages 2672–2680, 2014.

[8] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018.

[9] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving Variational Inference with Inverse Autoregressive Flow. *arXiv:1606.04934*, June 2016.

References (cont.)

[10] D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[11] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[12] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015.

[13] L. Ruthotto and E. Haber. An introduction to deep generative modeling, 2021.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. pages 2234–2242, 2016.

[15] J. Stanczuk, C. Etmann, L. M. Kreusser, and C.-B. Schönlieb. Wasserstein gans work because they fail (to approximate the wasserstein distance), 2021.