Deep Generative Model: A Statistical Perspective

Young-geun Kim

Department of Statistics and Probability

STT 990 (Fall 2024)

MICHIGAN STATE UNIVERSITY

Outline

What are Generative Models?

- The term 'Generative Model' has been used in classification since before the emergence of deep generative models.
- Let X and Y represent observations and labels, respectively, in a classification problem.
  1. Generative Model: Models $p(X, Y)$, typically by modeling $p(X|Y)$ and $p(Y)$ separately.
  2. Discriminative Model: Models $p(Y|X)$ only and uses it directly.
- Since the generative model learns $p(X, Y)$, it can generate data, e.g., by first sampling $Y \sim \text{Ber}(p)$ and then sampling $X|Y \sim N(\mu_Y, \sigma_Y^2)$ to synthesize (X, Y) pairs.

What are Generative Models?

- The advent of high-dimensional and large-scale data has increased the richness of information within the data, even without human-annotated labels, emphasizing the need for advanced statistical methods.
- Recent advancements have enabled the learning and generation of complex data. These models are now commonly referred to as Generative AI or Generative Models.
- Similar to traditional generative models used in classification, they learn the joint distribution of all observed variables, $\vec{X}$:

$$p(\vec{X}). \tag{1}$$

  This capability allows them to generate new data samples.

What are Generative Models?

- In this talk, the term ‘Generative Model’ refers to statistical models that learn the distribution of observations either without human-annotated labels or with auxiliary information.[1]
- Deep generative models have shown remarkable performance as:
  1. simulators by generating realistic data,
  2. dimension reduction tools by extracting low-dimensional representations,
  3. inference tools by translating observations to other domains.

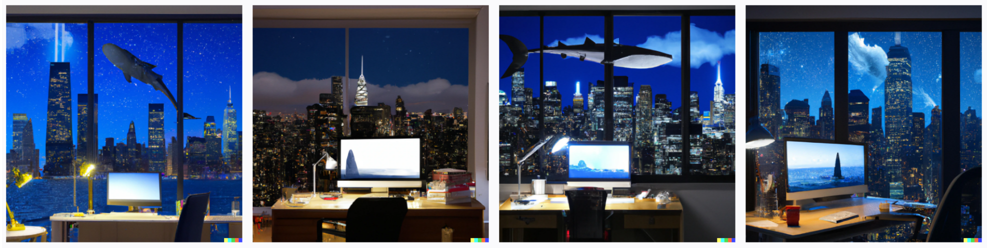
[1] For example, demographic information in medical data or timestamps in temporal data.

Application: Image Generation

- After training, generative models can synthesize realistic data without prior information such as age and race in face generation.
- These models can be used to augment data, generate privacy-free samples, and enhance virtual reality experiences.

The generated image is from `http://thispersondoesnotexist.com/`

Application: Text-to-Image Generation



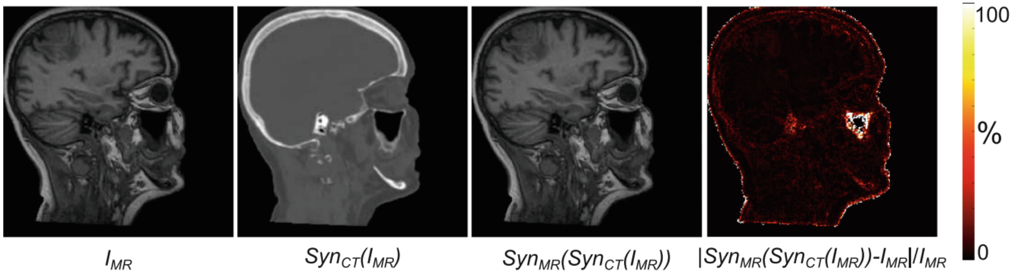
- Images are generated by DALL-E-2 using

  "There is a clean desk in the middle. Outside the window, a whale shark is swimming in the dark night sky above Manhattan."

- Generated images reflect semantic information in text descriptions.

Images are generated with DALL-E-2 (Ramesh et al., 2022).

Application: Cross Modality Transfer


I_{MR}
$Syn_{CT}(I_{MR})$
$Syn_{MR}(Syn_{CT}(I_{MR}))$
$|Syn_{MR}(Syn_{CT}(I_{MR}))-I_{MR}|/I_{MR}$
100
%
0

- Generative models are also useful in imputing missing modalities, e.g., MR $\rightarrow$ CT, or enhancing data resolution.

Images are from Wolterink et al. (2017).

Challenges in Deep Generative Models

- **High-dimensional Data**: For data like 4K-resolution color images, the dimensionality is $3,840 \times 2,160 \times 3 (\approx 24M)$.
- **Complex Structure**: Image, video, audio, and language data exhibit complex structures.

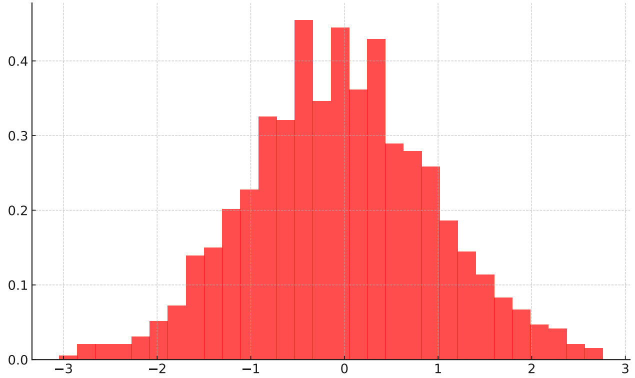
Challenge in Deep Generative Models

- How to model the distribution of high-dimensional data efficiently?
- How to evaluate the generated data and train the model distribution?

Example of generated images, edited from Li et al. (2024)

How to Learn Distributions: Toy Example

- Let's begin with a basic example. Assume we observed n univariate samples $x_1, \ldots, x_n$ having the following histogram:



How to Learn Distributions: Toy Example

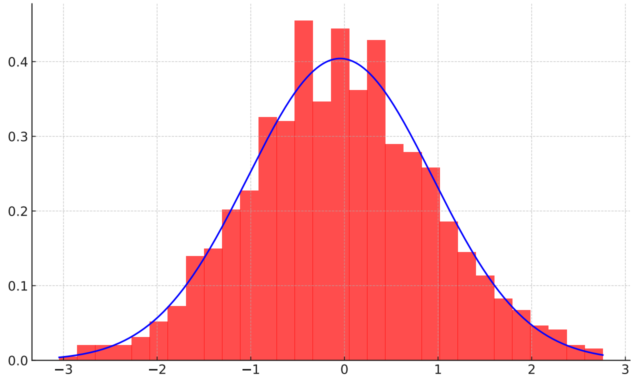
- How to learn the distribution where x_i comes from?
- One way is to model it as Gaussian and find the optimal mean and std parameters.
- $p_\theta(x) := (2\pi\sigma^2)^{-1/2} \exp\left((x-\mu)^2/(2\sigma^2)\right)$ where $\theta = (\mu, \sigma^2)^T \in \mathbb{R} \times \mathbb{R}^+$.[2]

Q: What are good evaluation criteria for optimality? How do we determine which parameter values are superior?

[2] For brevity, parameter vectors are denoted without vector symbols if there is no confusion.

How to Learn Distributions: Toy Example

- $l_n(\theta) := \sum_{i=1}^n \log p_\theta(x_i) = -(n/2)\log\sigma^2 - \sum_{i=1}^n (x_i-\mu)^2/(2\sigma^2) - (n/2)\log 2\pi$
- $\arg\max_\theta l_n(\theta) = (\hat{\mu}_n, \hat{\sigma_n^2})^T = \left(\bar{x}, n^{-1}\sum_{i=1}^n (x_i-\bar{x})^2\right)^T$



How to Learn Distributions: Toy Example

- We can explain the maximum likelihood principle using Kullback-Leibler (KL) divergence.
- Note that $n^{-1} l_n(\theta) = \int \Big(\log p_\theta(x) \Big) p_n(x) dx$ where $p_n := n^{-1} \sum_{i=1}^{n} \delta_{x_i}$. It implies

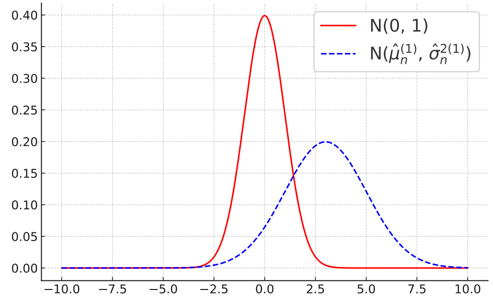
$$\begin{aligned} n^{-1} l_n(\theta) &= \int \Big(\log p_\theta(x) / p_n(x) \Big) p_n(x) dx + \int \Big(\log p_n(x) \Big) p_n(x) dx \\ &= -\mathrm{KL}(p_n \| p_\theta) + \mathrm{C} \end{aligned} \tag{2}$$

  where C is a constant w.r.t. θ.[3]
- Thus, MLEs are minimizers of the KL divergence between the empirical measure and the model density.

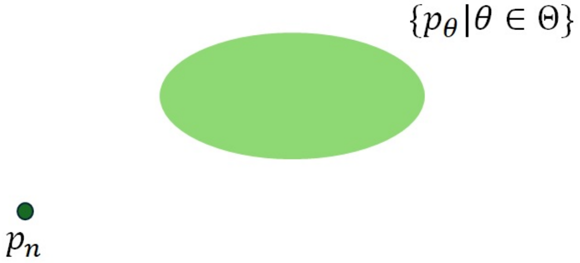
[3]Formally, $\mathrm{KL}(p||q) := \int \log \Big(\mathbb{P}(dx)/\mathbb{Q}(dx) \Big) \mathbb{P}(dx)$ where $\mathbb{P} \ll \mathbb{Q}$.

How to Learn Distributions: Toy Example

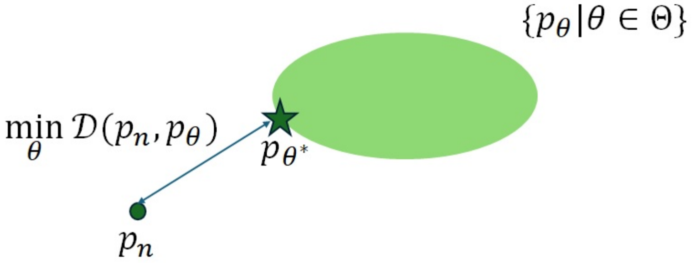
- There is no closed-form expression for the MLEs when addressing complex data and models.
- We usually run iterative algorithms to approximate optimal parameters.



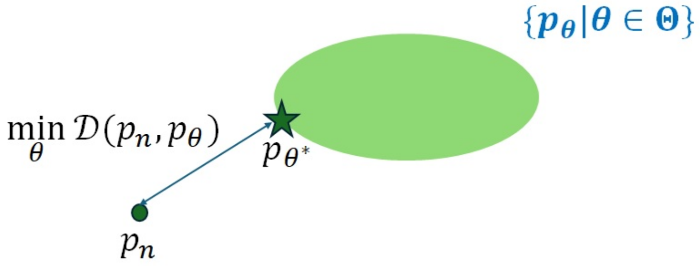
Key Elements in Generative Model



Key Elements in Generative Model

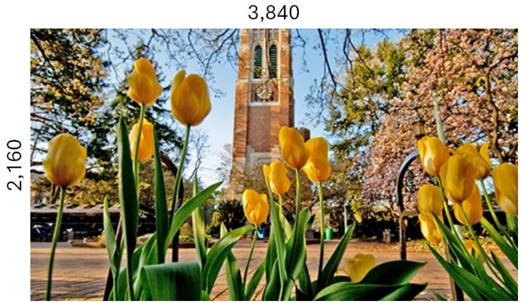

$\{p_\theta | \theta \in \Theta\}$
$\min_\theta \mathcal{D}(p_n, p_\theta)$
p_{θ^*}
p_n

Key Elements in Generative Model


$\{p_\theta | \theta \in \Theta\}$
$\min_\theta \mathcal{D}(p_n, p_\theta)$
p_{θ^*}
p_n

- Key elements to learn generative models include
  1. **Model Class**: Graphical models are frequently used to reflect domain knowledge
  2. **Statistical Distance**: Statistical distances measure differences between distributions to identify optimal generative models

Model Class



- For example, $\vec{X} := (X_1, \ldots, X_m)^T \in \mathcal{X}^m$ represents a m-dimensional random vector for the 4K-resolution color images where $m \approx 24M$.[4]
- Each color channel value is discrete, ranging from 0 to 255 ($|\mathcal{X}| = 256$).

Q: How to model $p(X_1 = x_1, \ldots, X_m = x_m)$?

[4]From now on, x_i represents the realization of the i-th element in $\vec{X}$.

Examples of Model Classes

1. **Multivariate Categorical Distribution**: Without any domain knowledge, we can introduce parameters for each realization, e.g., $p(X_1 = 0, \ldots, X_m = 0)$.

- It can express all the distributions defined on the data domain. However, the number of parameters is huge, about $|\mathcal{X}|^m$ (approximately 10^{60M}).

2. **Degenerated Model**: When $(X_2, \ldots, X_m)$ are (known) deterministic functions of X_1, introducing parameters for the marginal distribution $p(X_1)$ is sufficient.

- The number of parameters is small, about $|\mathcal{X}|$, and invariant to the data dimension. However, the model class is significantly reduced.

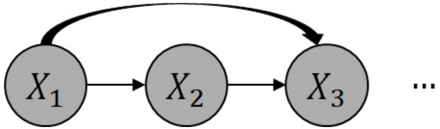
Building appropriate models that reflect domain knowledge of dependency structure is important

Model Classes by Dependency Structure

- We can express $p(X_1, \ldots, X_m)$ as a product of conditional distributions:

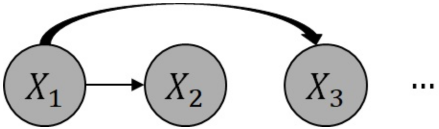
$$p(X_1)p(X_2|X_1)\ldots p(X_m|X_1, X_2, \ldots, X_{m-1}). \quad (3)$$

Many dependency structures can be represented by a directed graph with m nodes $X_1, \ldots, X_m$ and $(m-1)m/2$ directed edges, such as $X_1 \to X_2, \ldots, X_{m-1} \to X_m$.

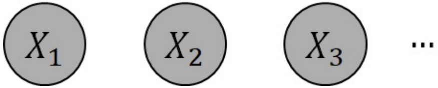


- For example, **1. Multivariate Categorical Distribution** corresponds to the graph using the all nodes and directed edges.

Model Classes by Dependency Structure

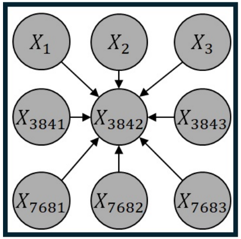


- **2. Degenerated Model**: This model is a special case of the above graph where $p_\theta(X_1, \dots, X_m) = p_\theta(X_1)p_\theta(X_2|X_1)\dots p_\theta(X_m|X_1)$.



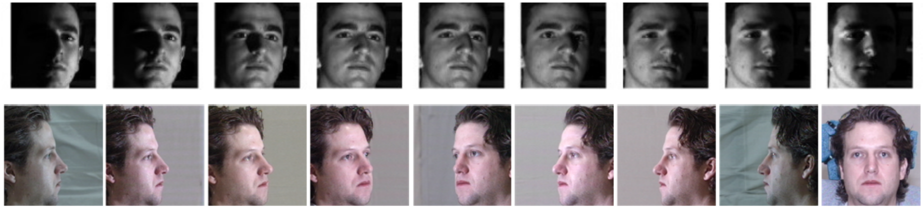
3. **Multivariate Independent Categorical Distribution**: This model assumes the (mutual) independency among variables, using $p_\theta(X_1, \dots, X_m) = p_\theta(X_1)\dots p_\theta(X_m)$. It requires $|\mathcal{X}|\ m$ parameters, but its assumption is strong.

Model Classes by Dependency Structure



4. **Spatial Model**: The value of the center pixel (e.g., X_{3842}) depends only on adjacent pixels (e.g., $X_1, \ldots, X_{7683}$), making it conditionally independent of all other pixels.
   - The number of parameters is about $|\mathcal{X}|^{\#\text{ of adjacent pixels } + 1} \times m$. Assuming translation invariance reduces it to approximately $|\mathcal{X}|^{\#\text{ of adjacent pixels } + 1}$.

Model Classes by Dependency Structure

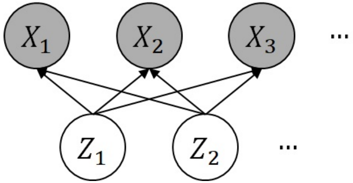


5. **Latent Variable Model**: There are latent factors $\vec{Z} := (Z_1, \ldots, Z_r)^T$, typically consisting of independent components, that are mixed to generate data, e.g., $\vec{X} = A\vec{Z}$ in Independent Component Analysis.

- The age, size, and location of eyes, light source location, and camera angle are examples of latent factors.

The top and bottom images are from the Extended Yale-B (Georghiades et al., 2001) and Multi-pie (Gross et al., 2010) datasets, respectively.

Model Classes by Dependency Structure

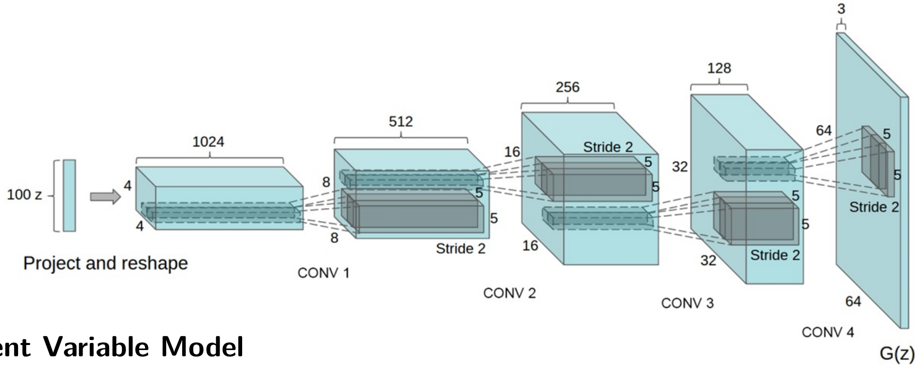


5. Latent Variable Model

$$\begin{aligned} p_\theta(X_1, \ldots, X_m) &= \int \Big(p_\theta(X_1, \ldots, X_m | \vec{Z} = \vec{z}) \Big) p(\vec{Z} = \vec{z}) \, d\vec{z} \\ &= \int \Big(p_\theta(X_1 | \vec{Z} = \vec{z}) \cdots p_\theta(X_m | \vec{Z} = \vec{z}) \Big) \prod_{i=1}^{r} p(Z_i = z_i) \, d\vec{z} \end{aligned} \tag{4}$$

- When we have discrete $\vec{Z}$, the number of parameters is approximately $\Big(|\mathcal{X}| \cdot |\mathcal{Z}|^r \Big) m$.

Model Classes by Dependency Structure

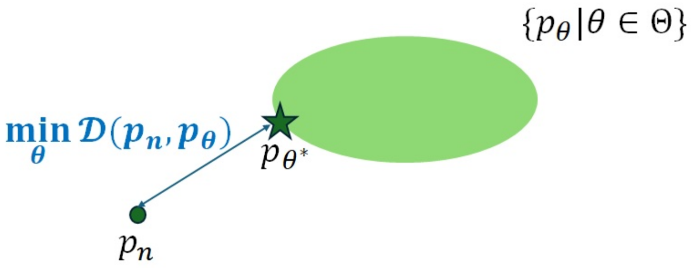


5. Latent Variable Model

- Deep generative models are predominantly based on latent variable models.
- The conditional distributions $p_\theta(\vec{X}|\vec{Z})$ are usually modeled as parametric family distributions, e.g., $N(\boldsymbol{\mu}_{\vec{X}|\vec{Z}}(\vec{Z}), \boldsymbol{\Sigma}_{\vec{X}|\vec{Z}}(\vec{Z}))$, which further reduce the number of parameters.

An example neural network for deep generative model from Radford (2015).

Key Elements in Generative Model



$\{p_\theta | \theta \in \Theta\}$
$\min_\theta \mathcal{D}(p_n, p_\theta)$
p_{θ^*}
p_n

- Key elements to learn generative models include
  1. **Model Class**: Graphical models are frequently used to reflect domain knowledge
  2. **Statistical Distance**: Statistical distances measure differences between distributions to identify optimal generative models

Examples of Statistical Distances

- Deep generative models learn p_n by minimizing $\mathcal{D}(p_n, p_\theta)$, where $\mathcal{D}$ denotes a statistical distance.
- The effectiveness of $\mathcal{D}$ varies depending on the type of data and the specific algorithm implementation. Each $\mathcal{D}$ necessitates different model classes and corresponding loss functions.
- Popular choices of $\mathcal{D}$ include:
  1. f-divergence
  2. Integral Probability Metric
  3. Wasserstein Distance[5]
  4. Fisher Divergence

[5] Named after "Leonid Vaseršteĭn", though "Wasserstein" is more commonly used in English publications.

Examples of Statistical Distances

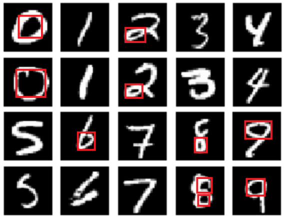
1. f-**divergence**:

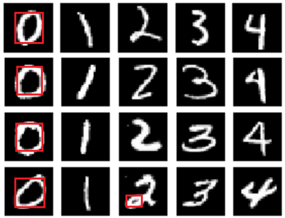
- The f-divergences (Rényi, 1961) are expectations of density ratios mapped by convex functions $f : \mathbb{R}^+ \to \mathbb{R}$, satisfying $f(1) = 0$. They can be expressed as:

$$\mathcal{D}_f(p||q) := \int f\left(\frac{p(\vec{x})}{q(\vec{x})}\right) q(\vec{x})\, d\vec{x} \tag{5}$$

- The $\mathrm{KL}(p_n||p_\theta)$ equals $\mathcal{D}_f(p_n||p_\theta)$ when $f(u) = u \log u$.
  Proof:
  $\mathcal{D}_f(p_n||p_\theta) = \int \Big(p_n(\vec{x})/p_\theta(\vec{x})\Big) \log \Big(p_n(\vec{x})/p_\theta(\vec{x})\Big) p_\theta(\vec{x}) d\vec{x} = \int \log \Big(p_n(\vec{x})/p_\theta(\vec{x})\Big) p_n(\vec{x}) d\vec{x}$.
- Thus, all maximum likelihood methods target minimizing this specific f-divergence.

Examples of Statistical Distances


Real


Generated

2. **Integral Probability Metric**: The integral probability metrics (IPMs, Müller, 1997) between distributions are the largest difference between their summary statistics.

- For example, when we use the number of circles in images as summary statistics, the difference is 0.5 (real) $-$ 0.25 (generated) $=$ 0.25.

Image source: MNIST (Deng, 2012)

Examples of Statistical Distances

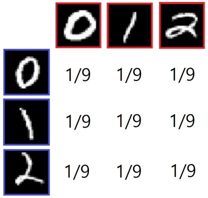
2. **Integral Probability Metric**:

- We can consider many summary statistics together to precisely compare distributions.
- The IPMs can be expressed as:

$$\gamma_{\mathcal{F}}(\mathbb{P}, \mathbb{Q}) := \sup_{f \in \mathcal{F}} \left| \int f(\vec{x})\, d\mathbb{P}(\vec{x}) - \int f(\vec{x})\, d\mathbb{Q}(\vec{x}) \right| \tag{6}$$

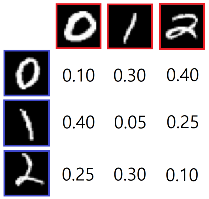
  where $\mathcal{F}$ is a class of real-valued functions, and $\mathbb{P}$ and $\mathbb{Q}$ are probability measures.
- Examples of $\mathcal{F}$ include the class of all 1-Lipschitz continuous functions, and functions from reproducing kernel Hilbert space (RKHS).

Examples of Statistical Distances



	0	1	2
0	1/9	1/9	1/9
1	1/9	1/9	1/9
2	1/9	1/9	1/9

Joint Distribution



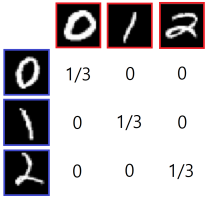
	0	1	2
0	0.10	0.30	0.40
1	0.40	0.05	0.25
2	0.25	0.30	0.10

Transportation Cost

3. **Wasserstein Distance**: Wasserstein distance (Monge, 1781; Kantorovich, 1960) represents the minimum expected transportation cost between two distributions.
   - For example, consider the above joint distribution. The transportation cost is calculated as:

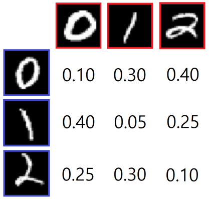
$$
\begin{aligned}
&(1/3) \times \big(0.10 \times (1/3) + 0.30 \times (1/3) + 0.40 \times (1/3)\big) \\
&+ (1/3) \times \big(0.40 \times (1/3) + 0.05 \times (1/3) + 0.25 \times (1/3)\big) \\
&+ (1/3) \times \big(0.25 \times (1/3) + 0.30 \times (1/3) + 0.10 \times (1/3)\big) = 0.24.
\end{aligned}
$$

Examples of Statistical Distances



	0	1	2
0	1/3	0	0
1	0	1/3	0
2	0	0	1/3

Joint Distribution



	0	1	2
0	0.10	0.30	0.40
1	0.40	0.05	0.25
2	0.25	0.30	0.10

Transportation Cost

3. Wasserstein Distance

- We can consider another joint distribution. The transportation cost is

$$
\begin{aligned}
&(1/3)\big(0.10(1) + 0.30(0) + 0.40(0)\big) \\
&+ (1/3)\big(0.40(0) + 0.05(1) + 0.25(0)\big) \\
&+ (1/3)\big(0.25(0) + 0.30(0) + 0.10(1)\big) = 0.08.
\end{aligned}
$$

Examples of Statistical Distances

3. **Wasserstein Distance**

- The p-**Wasserstein distance** can be expressed as

$$W_p(\mathbb{P}, \mathbb{Q}; d) := \left(\inf_{\pi \in \Pi(\mathbb{P}, \mathbb{Q})} \int d^p(\vec{x}, \vec{x}') d\pi(\vec{x}, \vec{x}') \right)^{1/p} \tag{7}$$

where $p \in [1, \infty)$ and $\Pi(\mathbb{P}, \mathbb{Q})$ is the set of all joint distributions whose marginals are $\mathbb{P}$ and $\mathbb{Q}$.

Examples of Statistical Distances

4. **Fisher Divergence**: Fisher divergence (Johnson, 2004; Hyvärinen, 2005) is the expected difference between the (Stein) scores (Liu et al., 2016)[6] of two distributions. It can be expressed as:

$$\mathrm{FD}(p \parallel q) = \int \left\| \nabla_{\vec{x}} \log p(\vec{x}) - \nabla_{\vec{x}} \log q(\vec{x}) \right\|^2 p(\vec{x})\, d\vec{x}. \tag{8}$$

- It is zero if and only if $p = q$.
  Proof:

$$\nabla_{\vec{x}} \log p(\vec{x}) = \nabla_{\vec{x}} \log q(\vec{x}) \implies p(\vec{x}) = Cq(\vec{x}) \tag{9}$$

and $C = 1$ because $\int p(\vec{x}) d\vec{x} = \int q(\vec{x}) d\vec{x} = 1$.

[6] The term 'score' here refers to the gradient w.r.t. realizations, which differs from usual terms in parametric family distributions.

Examples of Statistical Distances

4. **Fisher Divergence**: This is a useful measure for learning energy-based models (Teh et al., 2003), such as Boltzmann distributions:

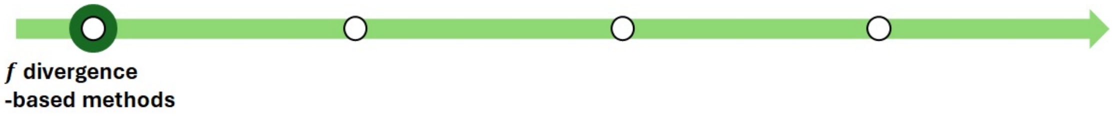
$$p_\theta(\vec{x}) = \frac{1}{C(\theta)} \exp(-E_\theta(\vec{x})) \tag{10}$$

where $C(\theta) := \int \exp(-E_\theta(\vec{x}))\, d\vec{x}$. In this case,

$$\nabla_{\vec{x}} \log p_\theta(\vec{x}) = -\nabla_{\vec{x}} E_\theta(\vec{x}) \tag{11}$$

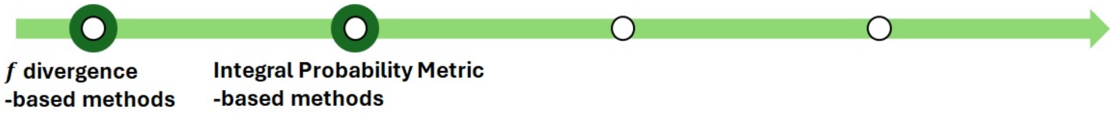
holds, and the normalizing constant disappears.

Chronicle of Deep Generative Model



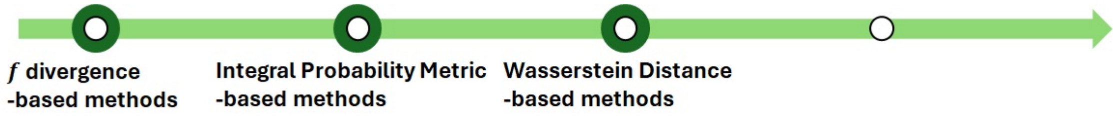
- Variational Autoencoder (VAE, Kingma and Welling, 2014)
- Generative Adversarial Network (GAN, Goodfellow et al., 2014)
- f-GAN (Nowozin et al., 2016)

Chronicle of Deep Generative Model



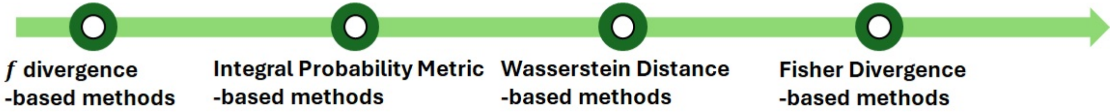
- Generative Moment Matching Networks (Li et al., 2015)
- Maximum Mean Discrepancy GANs (Li et al., 2017)
- Sobolev GANs (Mroueh et al., 2017)

Chronicle of Deep Generative Model


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods

- Wasserstein GANs (Arjovsky et al., 2017)
- Wasserstein GAN with gradient penalty (Gulrajani et al., 2017)
- Wasserstein Autoencoders (Tolstikhin et al., 2018)

Chronicle of Deep Generative Model


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods

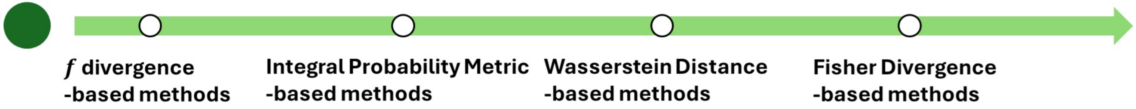
- Noise Conditional Score Networks (Song and Ermon, 2019)
- Denoising Diffusion Probabilistic Models (Ho et al., 2020)

Advanced Topics

- Asymptotic efficiencies according to statistical distances, e.g., minimax convergence rates of IPM-based generative models (Uppal et al., 2019).
- Methods for data domains other than images, e.g., generative pre-trained transformers (GPTs, Radford, 2018) and their variations for language data.
- Advanced graphical models reflecting domain knowledge, e.g., DALL-E (Ramesh et al., 2021) for generating images from text descriptions.

Outline

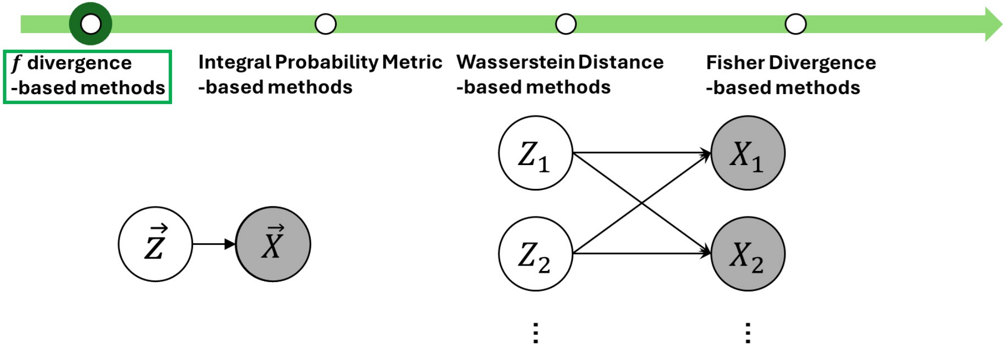
Limitation of Pre-Deep Generative Model


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods

$$\begin{tikzcd} X_1 \arrow[r] & X_2 \arrow[r] & X_3 & \cdots \end{tikzcd}$$

- Before the emergence of deep generative models, state-of-the-art methods typically employed Markov models, requiring extensive Markov chain Monte Carlo computations.

Emergence of f-Divergence-based Methods



f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods
$\vec{Z}$
$\vec{X}$
Z_1
Z_2
X_1
X_2

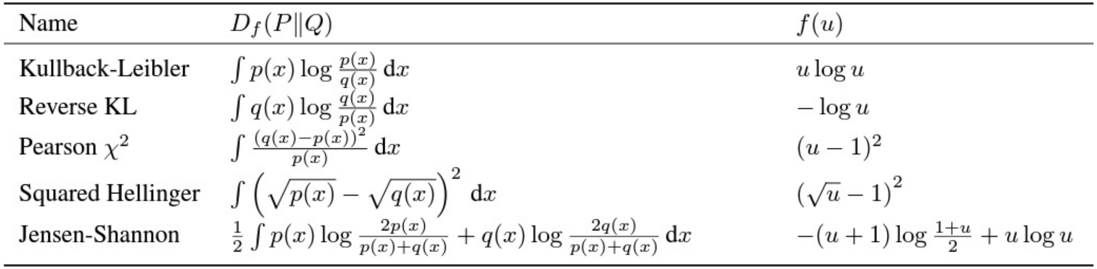
- A line of work introduced latent variable models, utilizing deep neural networks to model generator functions that mix latent variables to synthesize high-dimensional observations.
- VAEs and GANs are popular examples of these methods, specifically targeting the f-divergence, $\mathcal{D}_f(p_n \| p_\theta)$.

Recapping f-Divergence

- $\mathcal{D}_f(p||q) := \int f\left(p(\vec{x})/q(\vec{x})\right) q(\vec{x})\, d\vec{x}$ where $f : \mathbb{R}^+ \to \mathbb{R}$ is a convex function satisfying $f(1) = 0$. Examples include KL divergence, total variation distance, and Jensen-Shannon (JS) divergence:
- **KL divergence**: $\mathrm{KL}(p||q) := \int \log(p(\vec{x})/q(\vec{x}))p(\vec{x})d\vec{x}$
- **Total variation distance**: $\delta(p, q) := \frac{1}{2}\int \left|p(\vec{x}) - q(\vec{x})\right| d\vec{x}$
- **Jensen-Shannon divergence**:

$$\mathrm{JS}(p||q) := \frac{1}{2}\Big(\mathrm{KL}(p||\frac{p+q}{2}) + \mathrm{KL}(q||\frac{p+q}{2})\Big) \tag{12}$$

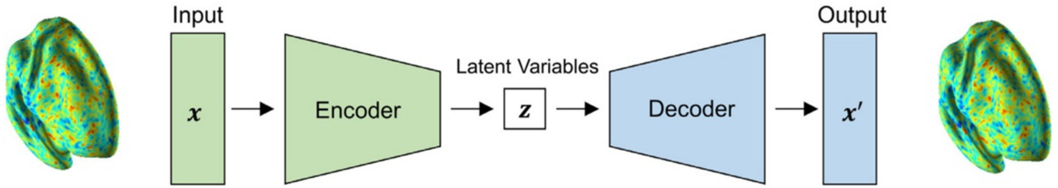
Recapping f-Divergence



Name	$D_f(P\|Q)$	$f(u)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x$	$u \log u$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \, \mathrm{d}x$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \, \mathrm{d}x$	$(u-1)^2$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \, \mathrm{d}x$	$\left(\sqrt{u}-1\right)^2$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \, \mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$

List of popular examples of f-divergences, edited from Nowozin et al. (2016).

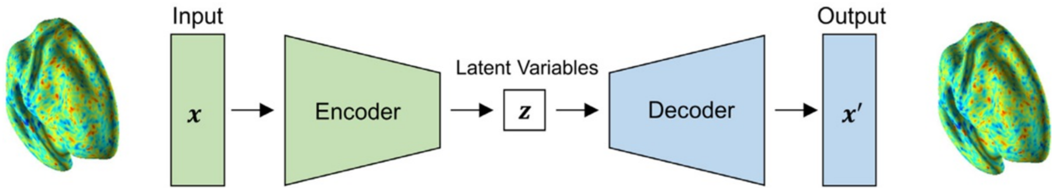
KL Divergence: Variational Autoencoder



1. **Variational Autoencoder**:
   - We first briefly review autoencoders (Bengio et al., 2006). Autoencoders (AEs) consist of pairs of encoders and decoders that efficiently reduce the dimensionality of data.
   - Encoders embed observations into a lower-dimensional space (referred to as ‘encoding’), while decoders map these encodings back to the original observation space (‘decoding’ or ‘reconstruction’).

Images are from Kim et al., 2021.

KL Divergence: Variational Autoencoder



- The prefix ‘auto’ is used because they autonomously learn to encode data in an unsupervised manner.
- Autoencoders (AEs) are trained by minimizing the difference between the original observations and their reconstructions, referred to as the ‘reconstruction error’.

Images are from Kim et al., 2021.

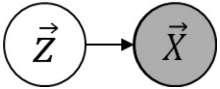
KL Divergence: Variational Autoencoder

- AEs are nonlinear extensions of Principal Component Analysis (Kramer, 1991; Plaut, 2018).
- Assuming the data $(\vec{x}_i)_{i=1}^n$ is centered, for a given dimension r, we define:

$$W^* \in \underset{W}{\arg\min} \left(n^{-1} \sum_{i=1}^{n} \|\vec{x}_i - WW^T\vec{x}_i\|^2 \right) \text{ subject to } W^TW = I_r. \quad (13)$$

  Here, $W^T\vec{x}_i$ represents the encoding process, and $WW^T\vec{x}_i$ represents the decoding.
- The W^* identifies optimal linear encoder and decoder pairs among symmetric AEs.
- The optimal embeddings $W^{*T}\vec{x}_i$ are the first r principal components up to orthogonal transformations.

KL Divergence: Variational Autoencoder

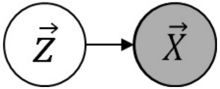


- Variational Autoencoders (VAEs, Kingma and Welling, 2014) model the data generation process using decoder networks:

$$\begin{aligned} p_\theta(\vec{z},\vec{x}) &:= p(\vec{z})p_\theta(\vec{x}|\vec{z}) \\ &= \Big(\prod_{i=1}^{r} p(z_i)\Big)\Big(\prod_{i=1}^{m} p_\theta(x_i|\vec{z})\Big). \end{aligned} \tag{14}$$

  where $p(\vec{z})$ is called the 'prior' distribution.
- Let $p_\theta(\vec{x}) := \int p_\theta(\vec{z},\vec{x})d\vec{z}$. Then, $p_\theta(\vec{z}|\vec{x})$ represents the 'posterior' distribution.

KL Divergence: Variational Autoencoder



- VAEs use $N(0, I)$ for $p(\vec{z})$ and $N(\boldsymbol{\mu}_{\vec{X}|\vec{Z}}(\vec{Z}), \boldsymbol{D}_{\vec{X}|\vec{Z}}(\vec{Z}))$ for $p_\theta(\vec{x}|\vec{z})$. Here, $\boldsymbol{\mu}_{\vec{X}|\vec{Z}}$ is defined as $(\mu_{X_1|\vec{Z}}, \ldots, \mu_{X_m|\vec{Z}})^T$ and $\boldsymbol{D}_{\vec{X}|\vec{Z}}$ as $\text{diag}(\sigma^2_{X_1|\vec{Z}}, \ldots, \sigma^2_{X_m|\vec{Z}})$, and all elements are outputs of neural networks parameterized by θ.
- In this context, the joint distribution is given by:

$$p_\theta(\vec{z}, \vec{x}) = \prod_{i=1}^{r} \left(\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z_i^2}{2}\right)\right) \prod_{i=1}^{m} \left(\frac{1}{\sqrt{2\pi\sigma^2_{X_i|\vec{Z}}(\vec{z})}} \exp\left(-\frac{(x_i - \mu_{X_i|\vec{Z}}(\vec{z}))^2}{2\sigma^2_{X_i|\vec{Z}}(\vec{z})}\right)\right). \tag{15}$$

KL Divergence: Variational Autoencoder

- Since $\vec{Z}$ is unobserved, VAEs target to maximize $p_\theta(\vec{x}) := \int p(\vec{z})p_\theta(\vec{x}|\vec{z})d\vec{z}$. However, the $p_\theta(\vec{x})$ does not have a closed-form expression.
- VAEs apply variational inference (Bishop, 2006), introducing encoder to approximate the posterior as $q_\phi(\vec{z}|\vec{x})$. They maximize evidence lower bound (ELBO), a lower bound of the evidence $\log p_\theta(\vec{x})$:

$$\begin{aligned}\mathrm{ELBO}(\theta, \phi; \vec{x}) &:= \log p_\theta(\vec{x}) - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z}|\vec{x})) \\ &= \int \Big(\log p_\theta(\vec{x}|\vec{z})\Big) q_\phi(\vec{z}|\vec{x})d\vec{z} - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p(\vec{z})).\end{aligned} \tag{16}$$

Proof: By Bayes' rule, the relation $p_\theta(\vec{x}) = p_\theta(\vec{x}|\vec{z})p(\vec{z})/p_\theta(\vec{z}|\vec{x})$ holds, implying $\log p_\theta(\vec{x}) - \log\Big(q_\phi(\vec{z}|\vec{x})/p_\theta(\vec{z}|\vec{x})\Big) = \log p_\theta(\vec{x}|\vec{z}) - \log\Big(q_\phi(\vec{z}|\vec{x})/p(\vec{z})\Big)$. Taking the expectation over $q_\phi(\vec{z}|\vec{x})$ concludes the proof.

KL Divergence: Variational Autoencoder

- VAEs maximize the average of the ELBO, which is equivalent to minimizing

$$-\int \text{ELBO}(\theta, \phi; \vec{x}) p_n(\vec{x}) d\vec{x}. \tag{17}$$

- Define $\theta^* \in \arg\min_\theta \Big(\min_\phi \big(-\int \text{ELBO}(\theta, \phi; \vec{x}) p_n(\vec{x}) d\vec{x} \big) \Big)$. Then, p_{θ^*} is a minimizer of $\text{KL}(p_n \| p_\theta)$.
- We assume that the encoder class $\{q_\phi | \phi \in \Phi\}$ is sufficiently flexible such that for any given θ, there exists a $\phi^*(\theta)$ where: $q_{\phi^*(\theta)}(\vec{z}|\vec{x}) = p_\theta(\vec{z}|\vec{x})$ (a.s. w.r.t. $p_n(\vec{x})$).

KL Divergence: Variational Autoencoder

Proof: By Equation (16),

$$\begin{aligned}
&\min_{\phi}\Big(-\int \mathrm{ELBO}(\theta,\phi;\vec{x})p_n(\vec{x})d\vec{x}\Big)\\
&=\min_{\phi}\Bigg(-\int\Big(\log p_\theta(\vec{x})-\mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z}|\vec{x}))\Big)p_n(\vec{x})d\vec{x}\Bigg)\\
&=\mathrm{KL}(p_n||p_\theta)+\min_{\phi}\int \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z}|\vec{x}))p_n(\vec{x})d\vec{x}+C.
\end{aligned}$$

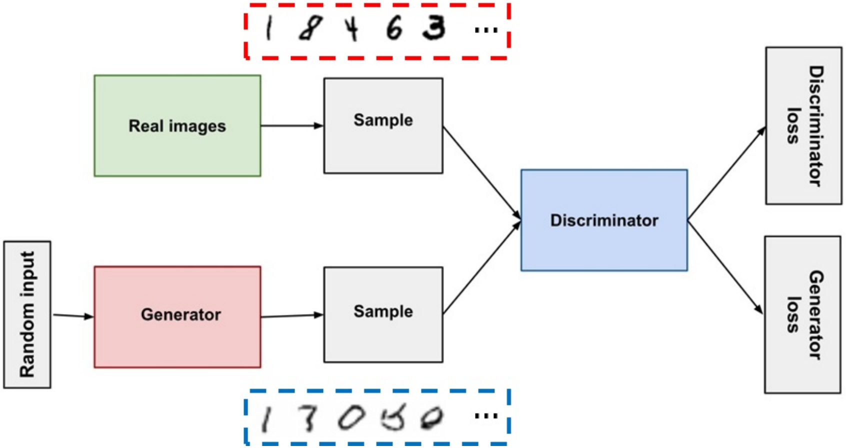
Thus, $\min_{\phi}\Big(-\int \mathrm{ELBO}(\theta,\phi;\vec{x})p_n(\vec{x})d\vec{x}\Big)=-\int \mathrm{ELBO}(\theta,\phi^*(\theta);\vec{x})p_n(\vec{x})d\vec{x}=\mathrm{KL}(p_n||p_\theta)$ up to a constant addition.

JS Divergence: Generative Adversarial Network

2. **Generative Adversarial Network**: Generative Adversarial Networks (GANs; Goodfellow et al., 2014) introduce adversarial learning through two networks: a generator and a discriminator.

- The generator specifies the same graphical model used in VAEs, but $\vec{x} = G_\theta(\vec{z})$ where G is a neural network, i.e., $p_\theta(\vec{x}|\vec{z})$ degenerates to a single point.
- The discriminator is a binary classifier designed to differentiate between real data and synthetic data produced by the generator.

JS Divergence: Generative Adversarial Network



Images were edited from https://developers.google.com/machine-learning/gan/gan_structure and https://github.com/MorvanZhou/mnistGANs.

JS Divergence: Generative Adversarial Network

- The adversarial learning process involves alternately maximizing and minimizing the negative cross-entropy loss:

$$V(\theta, \phi) := \int \Big(\log D_\phi(\vec{x}) \Big) p_n(\vec{x})\, d\vec{x} + \int \Big(\log(1 - D_\phi(\vec{x})) \Big) p_\theta(\vec{x})\, d\vec{x}. \tag{18}$$

- This process can be viewed as a two-player minimax game where the goal is to find

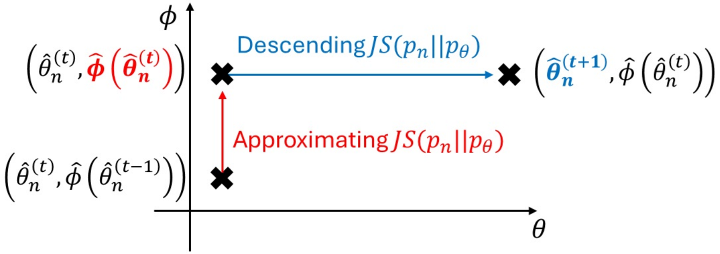
$$\theta^* \in \arg\min_\theta \Big(\max_\phi V(\theta, \phi) \Big). \tag{19}$$

- The adversarial training consists of repeated cycles of approximating and minimizing the JS divergence:

$$\max_\phi V(\theta, \phi) = \mathrm{JS}(p_n \parallel p_\theta) \tag{20}$$

up to a constant addition and sign-preserving multiplication. Thus, p_{θ^*} is the minimizer of $\mathrm{JS}(p_n \parallel p_\theta)$.

JS Divergence: Generative Adversarial Network


ϕ
$\left(\hat{\theta}_n^{(t)}, \hat{\boldsymbol{\phi}}\left(\hat{\boldsymbol{\theta}}_n^{(t)}\right)\right)$
Descending $JS(p_n||p_\theta)$
$\left(\hat{\boldsymbol{\theta}}_n^{(t+1)}, \hat{\phi}\left(\hat{\theta}_n^{(t)}\right)\right)$
Approximating $JS(p_n||p_\theta)$
$\left(\hat{\theta}_n^{(t)}, \hat{\phi}\left(\hat{\theta}_n^{(t-1)}\right)\right)$
θ

- The two adversarial networks, the discriminator and the generator, are trained alternately:

1. Given $\hat{\theta}_n^{(t)}$, update the discriminator to obtain $\hat{\phi}(\hat{\theta}_n^{(t)})$ by maximizing $V(\hat{\theta}_n^{(t)}, \phi)$.
2. Given $\hat{\phi}(\hat{\theta}_n^{(t)})$, update the generator to obtain $\hat{\theta}_n^{(t+1)}$ by minimizing $V(\theta, \hat{\phi}(\hat{\theta}_n^{(t)}))$.
3. Repeat the above processes.

JS Divergence: Generative Adversarial Network

- Again, $\max_{\phi} V(\theta, \phi) = \mathrm{JS}(p_n \parallel p_\theta)$ up to trivial transformations. We assume that the discriminator class $\{D_\phi \mid \phi \in \Phi\}$ is sufficiently flexible such that for any given θ, there exists a $\phi^*(\theta)$ where:

$$D_{\phi^*(\theta)}(\vec{x}) = \frac{p_n(\vec{x})}{p_n(\vec{x}) + p_\theta(\vec{x})}. \tag{21}$$

Proof: $V(\theta, \phi) = \int \Big(\log D_\phi(\vec{x}) p_n(\vec{x}) + \log(1 - D_\phi(\vec{x})) p_\theta(\vec{x}) \Big) d\vec{x}$, and the integrand is strictly concave w.r.t. $D_\phi(\vec{x})$. The first derivative of the integrand w.r.t. $D_\phi(\vec{x})$ is

$$-\frac{p_n(\vec{x}) + p_\theta(\vec{x})}{D_\phi(\vec{x})(1 - D_\phi(\vec{x}))} \left(D_\phi(\vec{x}) - \frac{p_n(\vec{x})}{p_n(\vec{x}) + p_\theta(\vec{x})} \right), \tag{22}$$

implying that $V(\theta, \phi)$ is maximized when Equation (21) holds. This implies

$$\max_{\phi} V(\theta, \phi) = V(\theta, \phi^*(\theta)) = 2\mathrm{JS}(p_n \parallel p_\theta) - \log 4. \tag{23}$$

f-Divergence: f-GAN

3. ***f*-GAN**:

- We have reviewed the following relationship in GANs, which holds up to a constant addition and sign-preserving multiplication: Using discriminator networks parameterized with ϕ,

$$\mathrm{JS}(p_n \parallel p_\theta) \approx V(\theta, \hat{\phi}_n(\theta)).$$

- Nowozin et al. (2016) generalized the concept of using auxiliary networks to approximate other f-divergences.
- The key idea is to introduce the convex conjugate function (or Fenchel conjugate, Hiriart-Urruty and Lemaréchal, 2004) to derive variational estimations of f-divergences.

f-Divergence: f-GAN

- We denote the convex conjugates of functions f by $f^*(t) := \sup_u \{ut - f(u)\}$.
- The f^* relates f and its subgradients. When f is convex and differentiable[7], the following properties hold:
  1. f^* is also convex and differentiable.
  2. Duality holds, i.e., $(f^*)^* = f$.
  3. The relation $f(u) + f^*(t) = ut$ holds if and only if $t = f'(u)$.
- When f' is invertible, $f^*(t) = (f')^{-1}(t)t - (f \circ f'^{-1})(t)$.

[7] For more general functions, check Hiriart-Urruty and Lemaréchal (2004).

f-Divergence: f-GAN

- By duality and the definition of supremum, we have the following variational formulation:

$$\begin{aligned}\mathcal{D}_f(p_n||p_\theta) &= \int \sup_t \left\{ \frac{p_n(\vec{x})}{p_\theta(\vec{x})} t - f^*(t) \right\} p_\theta(\vec{x}) d\vec{x} \\ &\geq \sup_{T_\phi} \left(\int T_\phi(\vec{x}) p_n(\vec{x}) d\vec{x} - \int f^*(T_\phi(\vec{x})) p_\theta(\vec{x}) d\vec{x} \right).\end{aligned} \tag{24}$$

- We assume that f is differentiable and that $\{T_\phi | \phi \in \Phi\}$ is sufficiently flexible such that for any given θ, there exists $\phi^*(\theta)$ where:

$$T_{\phi^*(\theta)}(\vec{x}) = f'(p_n(\vec{x})/p_\theta(\vec{x})). \tag{25}$$

This satisfies the equality condition of Equation (24).

f-Divergence: f-GAN

- Define $F(\theta, \phi) := \int T_\phi(\vec{x}) p_n(\vec{x})\, d\vec{x} - \int f^*\Big(T_\phi(\vec{x})\Big) p_\theta(\vec{x})\, d\vec{x}$ and $\theta^* := \arg\min_\theta \Big(\max_\phi F(\theta, \phi)\Big)$.
- Then, $\max_\phi F(\theta, \phi) = F(\theta, \phi^*(\theta)) = \mathcal{D}_f(p_n \| p_\theta)$. Thus, p_{θ^*} is a minimizer of the f-divergence.
- **Example 1 (KL divergence)**: Let $f(u) = u \log u$ and $f^*(t) = \exp(t-1)$. We can express $F(\theta, \phi)$ as follows:

$$F(\theta, \phi) = \int T_\phi(\vec{x}) p_n(\vec{x})\, d\vec{x} - \int \exp\Big(T_\phi(\vec{x}) - 1\Big) p_\theta(\vec{x})\, d\vec{x}. \tag{26}$$

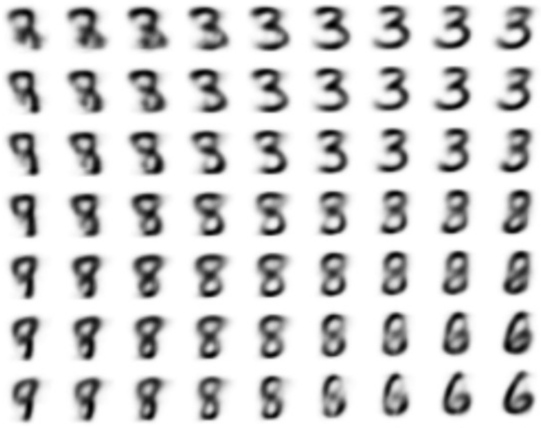
f-Divergence: f-GAN

- **Example 2 (JS divergence)**: Let $f(u) = -(u+1)\log\frac{1+u}{2} + u\log u$ and $f^*(t) = -\log(2-\exp(t))$. We can express $F(\theta,\phi)$ as follows:

$$F(\theta,\phi) = \int T_\phi(\vec{x})p_n(\vec{x})\,d\vec{x} + \int \log\left(2 - \exp\left(T_\phi(\vec{x})\right)\right)p_\theta(\vec{x})\,d\vec{x}. \tag{27}$$

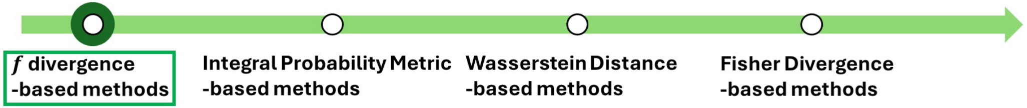
- GANs are special cases of f-GANs. When we model $T_\phi(\vec{x}) = \log D_\phi(\vec{x}) + \log 2$, $F(\theta,\phi) = V(\theta,\phi) + \log 4$, and $T_{\phi^*(\theta)}(\vec{x}) = \log D_{\phi^*(\theta)}(\vec{x}) + \log 2 = \log\frac{p_n(\vec{x})}{p_n(\vec{x})+p_\theta(\vec{x})} + \log 2$ hold.

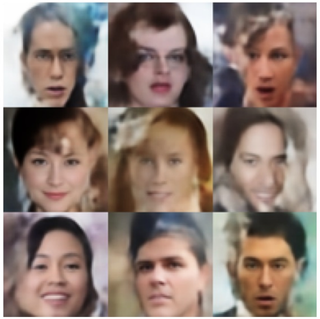
Generation Results: Latent Manifold Learned by VAEs

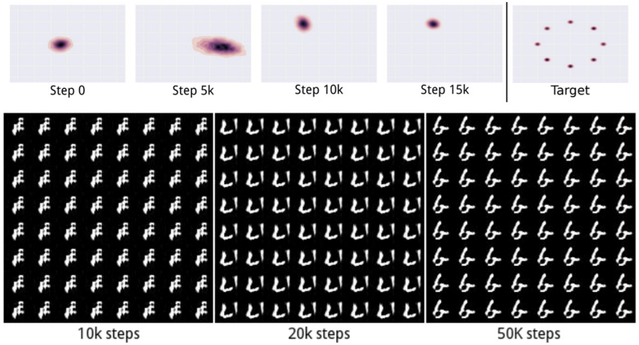


Images are edited from Kingma and Welling (2014).

Limitation of f Divergence-based Methods


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods


(a) Blurry generation results from VAEs



(b) Mode collapse results from GANs

Images are edited from Tolstikhin et al. (2018) and Metz et al. (2017).

Limitations of f-Divergence-Based Methods

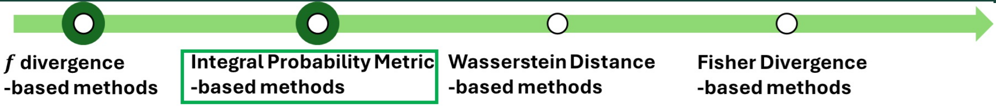
f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods

- The mode collapse phenomenon in GANs demonstrates that the p_θ fails in capturing the support of p_n.
- Several works have criticized f-divergence,

$$\mathcal{D}_f(p_n \| p_\theta) = \int f\left(\frac{p_n(\vec{x})}{p_\theta(\vec{x})}\right) p_\theta(\vec{x})\, d\vec{x},$$

  pointing out that it is based on the density ratio p_n/p_θ, and this dependency may be a reason for the observed failures in f-divergence-based methods.

Emergence of IPM and Wasserstein Distances-Based Methods

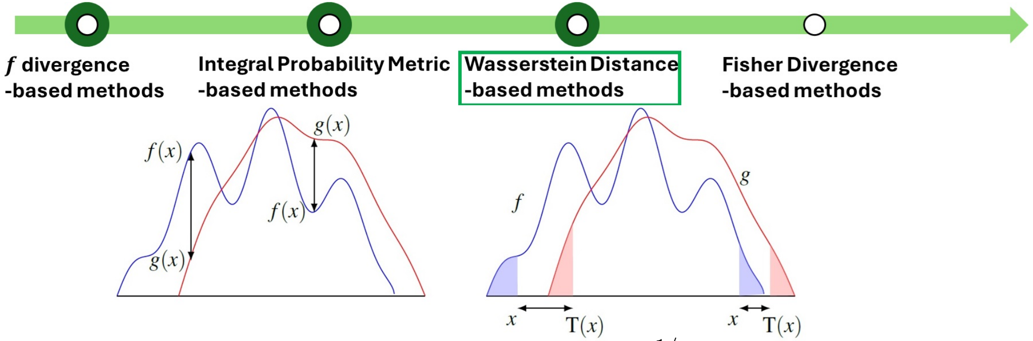

f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods

- As an alternative to density ratios, a line of work has proposed focusing on discrepancy measures that are effective regardless of the differences between the supports of p_n and p_θ.
- For example, Generative Moment Matching Networks (Li et al., 2015) aim to minimize:

$$\| \int \varphi(\vec{x}) d\mathbb{P}_n(\vec{x}) - \int \varphi(\vec{x}) d\mathbb{P}_\theta(\vec{x}) \|^2 \tag{28}$$

where the integrated terms $\varphi(\vec{x})$ represent vectors of finite moments, e.g., $\varphi(x) = (c, \sqrt{2c}x, x^2)^T$ in the univariate case with second-order moments.
- This loss function is a special case of integral probability metrics, $\gamma_{\mathcal{F}}(p_n, p_\theta)$, where $\mathcal{F}$ denotes a set of summary statistics functions, such as moments.

Emergence of IPM and Wasserstein Distances-based Methods


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods
$f(x)$
$g(x)$
$g(x)$
$f(x)$
f
g
x
$\mathrm{T}(x)$
x
$\mathrm{T}(x)$

- Another line of work has targeted $\left(\int d^p(\vec{x}, T(\vec{x}))\, d\vec{x}\right)^{1/p}$ where T transports data points from the initial distribution to the target distribution.
- This concept can be formulated as minimizing Wasserstein distances $W_p(p_n, p_\theta)$.

Images are edited from Santambrogio (2015).

Recapping Integral Probability Metric

- The IPMs can be expressed as:

$$\gamma_{\mathcal{F}}(\mathbb{P},\mathbb{Q}) := \sup_{f\in\mathcal{F}} \left| \int f(\vec{x})\, d\mathbb{P}(\vec{x}) - \int f(\vec{x})\, d\mathbb{Q}(\vec{x}) \right|$$

where $\mathcal{F}$ is a class of real-valued functions, and $\mathbb{P}$ and $\mathbb{Q}$ are probability measures.

Recapping Integral Probability Metric

- **Total Variation Distance**: The total variation distance, $\delta(p,q) = \frac{1}{2}\int \left| p(\vec{x}) - q(\vec{x}) \right| d\vec{x}$, has an alternative expression:

$$\sup_{A \in \mathcal{A}} \left| \mathbb{P}(A) - \mathbb{Q}(A) \right| = \sup_{A \in \mathcal{A}} \left| \int I(\vec{x} \in A) d\mathbb{P}(\vec{x}) - \int I(\vec{x} \in A) d\mathbb{Q}(\vec{x}) \right|$$

  where $\mathcal{A}$ is the corresponding σ-algebra. Thus, the total variation is the IPM using the set of indicator functions for all events.

- **Earth Mover's Distance**: When $\mathcal{F}$ consists of all 1-Lipschitz continuous functions, $\gamma_{\mathcal{F}}(\mathbb{P}, \mathbb{Q})$ corresponds to the Earth mover's distance (or 1-Wasserstein distance), a special case of Wasserstein distances. Further details will be discussed in the subsequent subsection on Wasserstein distances.

Recapping Integral Probability Metric

- **Maximum Mean Discrepancy (MMD)**: We denote the kernel mean by $\mu_{\mathbb{P}}(\vec{x}) := \int k(\vec{x}', \vec{x}) d\mathbb{P}(\vec{x}')$. Then, the MMD is defined as the difference between kernel means in $\mathcal{H}$, the RKHS specified by k:

$$\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) := \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}.$$

MMD builds a kernel-based test statistic for a two-sample test:

$$H_0 : \mathbb{P} = \mathbb{Q} \text{ vs. } H_1 : \mathbb{P} \neq \mathbb{Q}.$$

- The MMD has important alternative representations:
  1. **IPM:** $\mathrm{MMD}_k(\mathbb{P}, \mathbb{Q}) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \left(\int f(\vec{x}) d\mathbb{P}(\vec{x}) - \int f(\vec{x}) d\mathbb{Q}(\vec{x}) \right)$.
  2. **Kernel function form:**

$$\begin{aligned} &\mathrm{MMD}_k^2(\mathbb{P}, \mathbb{Q}) \\ &= \int k(\vec{x}, \vec{x}') d\mathbb{P}(\vec{x}) d\mathbb{P}(\vec{x}') - 2 \int k(\vec{x}, \vec{y}) d\mathbb{P}(\vec{x}) d\mathbb{Q}(\vec{y}) + \int k(\vec{y}, \vec{y}') d\mathbb{Q}(\vec{y}) d\mathbb{Q}(\vec{y}'). \end{aligned} \tag{29}$$

MMD: Generative Moment Matching Network

1. **Generative Moment Matching Network (GMMN)**: GMMNs (Li et al., 2015) propose to use empirical estimators as loss functions to train generative models rather than introducing adversarial networks as in GANs.

- Given $(\vec{x}_i)_{i=1}^B$ and $(G_\theta(\vec{z}_i))_{i=1}^B$, minibatch samples of size B from $\mathbb{P}_n$ and $\mathbb{P}_\theta$ respectively, the minibatch-based empirical estimators for $\mathrm{MMD}_k^2(\mathbb{P}_n, \mathbb{P}_\theta)$ can be expressed as

$$\begin{aligned} &\frac{1}{B(B-1)} \sum_{i=1}^B \sum_{j \neq i}^B k(\vec{x}_i, \vec{x}_j) - \frac{2}{B^2} \sum_{i=1}^B \sum_{j=1}^B k(\vec{x}_i, G_\theta(\vec{z}_j)) \\ &+ \frac{1}{B(B-1)} \sum_{i=1}^B \sum_{j \neq i}^B k(G_\theta(\vec{z}_i), G_\theta(\vec{z}_j)). \end{aligned} \tag{30}$$

- GMMNs used a mixture of multiple Gaussian kernels with various bandwidth parameters.

MMD: Generative Moment Matching Network

- Minimizing $\mathrm{MMD}_k(\mathbb{P}_n, \mathbb{P}_\theta)$ can be interpreted as matching moments between $\mathbb{P}_n$ and $\mathbb{P}_\theta$.
- Let k be the kernel that defines the MMD, and let $\varphi(\vec{x})$[8] represent the corresponding kernel feature mapping, i.e.,
$$k(\vec{x}, \vec{x}') = \varphi(\vec{x})^\top \varphi(\vec{x}') \tag{31}$$
- For a univariate example, consider $k(x, x') = (xx' + c)^2$ for some $c > 0$. The feature mapping $\varphi(x) = (c, \sqrt{2c}x, x^2)^\top$ satisfies Equation (31). Kernels with higher degrees allow for covering higher-order moments.
- The loss of GMMNs, minibatch-based empirical estimators for (squared) MMD, can be expressed as
$$\|B^{-1}\sum_{i=1}^{B}\varphi(\vec{x}_i) - B^{-1}\sum_{i=1}^{B}\varphi(G_\theta(\vec{z}_i))\|^2. \tag{32}$$

[8]The symbol ϕ is more commonly used, but we use φ here to avoid confusion with parameters for auxiliary networks, e.g., the discriminator in GANs.

MMD: MMD GAN

2. **MMD GAN**:

- GMMNs face challenges in selecting effective kernels. MMD GANs (Li et al., 2017) overcome this limitation by introducing adversarial kernel learning.
- MMD GANs aim to target $\max_{k \in \mathcal{K}} \text{MMD}_k(\mathbb{P}_n, \mathbb{P}_\theta)$, where $\mathcal{K}$ is a class of kernel functions.
- To model an expressive class $\mathcal{K}$, MMD GANs employ a neural network E_ϕ to define $(k \circ E_\phi)(\vec{x}, \vec{x}') := k(E_\phi(\vec{x}), E_\phi(\vec{x}'))$, targeting:

$$\max_{\phi} \text{MMD}_{k \circ E_\phi}(\mathbb{P}_n, \mathbb{P}_\theta). \tag{33}$$

- The injectivity of E_ϕ is crucial to retain the important properties of MMDs with usual kernels. MMD-GANs incorporate an encoder architecture for E_ϕ, add a decoder, and introduce a reconstruction error-based penalty term to enforce the injectivity.

Other IPMs

3. **Methods using Other IPMs**:

- One of the main challenges in using IPMs,

$$\gamma_{\mathcal{F}}(\mathbb{P}_n, \mathbb{P}_\theta) := \sup_{f \in \mathcal{F}} \left| \int f(\vec{x})\, d\mathbb{P}_n(\vec{x}) - \int f(\vec{x})\, d\mathbb{P}_\theta(\vec{x}) \right|,$$

  lies in approximating the supremum over the function class $\mathcal{F}$.
- While MMD has a tractable representation that allows for the direct use of its empirical estimators, this is not the case for more general IPMs.
- Most methods targeting other IPMs employ neural networks to model elements within $\mathcal{F}$. Notably, Wasserstein GANs (Arjovsky et al., 2017) have become one of the most popular methods targeting the 1-Wasserstein distance.

Recapping Wasserstein Distance

- The p-**Wasserstein distance** can be expressed as

$$W_p(\mathbb{P},\mathbb{Q};d) := \left(\inf_{\pi\in\Pi(\mathbb{P},\mathbb{Q})}\int d^p(\vec{x},\vec{x}')\,d\pi(\vec{x},\vec{x}')\right)^{1/p}$$

  where $p\in[1,\infty)$, and $\Pi(\mathbb{P},\mathbb{Q})$ is the set of all joint distributions whose marginals are $\mathbb{P}$ and $\mathbb{Q}$.

- (Monge-Kantorovich transportation problem) Under some conditions, there exists a map T that satisfies
  1. $W_p(\mathbb{P},\mathbb{Q};d) = \left(\int d^p(\vec{x},T(\vec{x}))d\mathbb{P}(\vec{x})\right)^{1/p}$
  2. $\mathbb{P}(T(\vec{x})) = \mathbb{Q}(\vec{x})$[9]

  The map T is called the 'optimal transport map'.

[9] This can be expressed with the push-forward operation $T\#\mathbb{P}=\mathbb{Q}$.

Recapping Wasserstein Distance

- For example, when d is the Euclidean norm, W_p becomes the Mallows metric (Mallows, 1972), and has played an important role in deriving asymptotic properties of bootstrap estimators (Bickel and Freedman, 1981; Freedman, 1981).
- When $p = 1$, duality holds (Villani et al., 2009; Villani, 2021), which provides an IPM formulation:

$$W_1(\mathbb{P}, \mathbb{Q}; d) = \sup_{\|f\|_L \leq 1} \int f(\vec{x}) d\mathbb{P}(\vec{x}) - \int f(\vec{x}) d\mathbb{Q}(\vec{x}) \tag{34}$$

where $\|f\|_L := \max\{C \big| |f(\vec{x}) - f(\vec{x}')| \leq Cd(\vec{x}, \vec{x}')\}$ represents the Lipschitz constant of f.

Recapping Wasserstein Distance

- Wasserstein distances effectively quantify differences between high-dimensional distributions when their supports are in low-dimensional manifolds.

Example

(Example 1 in Arjovsky et al., 2017) Let $Z \sim U[0,1]$, $X = (0, Z)$, and $G_\theta(Z) = (\theta, Z)$.

- Intuitively, $\mathcal{D}(\mathbb{P}_{n=\infty}, \mathbb{P}_\theta)$ should decrease as θ vanishes.
  - $W_p(\mathbb{P}_{n=\infty}, \mathbb{P}_\theta; |\cdot|) = |\theta|$
  - $\mathrm{JS}(\mathbb{P}_{n=\infty} \parallel \mathbb{P}_\theta) = \log 2$ if $\theta \neq 0$ and 0 if $\theta = 0$
  - $\mathrm{KL}(\mathbb{P}_{n=\infty} \parallel \mathbb{P}_\theta) = \infty$ if $\theta \neq 0$ and 0 if $\theta = 0$
  - $\delta(\mathbb{P}_{n=\infty}, \mathbb{P}_\theta) = 1$ if $\theta \neq 0$ and 0 if $\theta = 0$

1-Wasserstein Distance: Wasserstein GAN

1. **Wasserstein GAN (WGAN)**: WGANs model the class of 1-Lipschitz continuous functions using neural networks, denoted by f_ϕ, with the goal of

$$\min_{\theta} \max_{f_\phi} \left(\int f_\phi(\vec{x})\, d\mathbb{P}_n(\vec{x}) - \int f_\phi(\vec{x})\, d\mathbb{P}_\theta(\vec{x}) \right). \tag{35}$$

- When the set $\{f_\phi \mid \phi \in \Phi\}$ perfectly approximates the set $\{f \mid \|f\|_L \leq 1\}$, Equation (35) equals to $\min_\theta W_1(\mathbb{P}_n, \mathbb{P}_\theta; d)$.
- The 1-Lipschitz continuity condition is sometimes relaxed to C-Lipschitz continuity for an arbitrary constant C. To enforce this, WGANs clip weights and biases in neural network layers during training.

p-Wasserstein Distance: Wasserstein Autoencoder

2. **Wasserstein Autoencoder (WAE)**: Tolstikhin et al. (2018) derived an alternative representation of the p-Wasserstein distance:

$$W_p(\mathbb{P}_n, \mathbb{P}_\theta; d) = \left(\inf_{\mathbb{Q}(\vec{z}|\vec{x}) : \int q(\vec{z}|\vec{x}) d\mathbb{P}_n(\vec{x}) = p(\vec{z})} \int d^p(\vec{x}, G_\theta(\vec{z})) d\mathbb{Q}(\vec{z}|\vec{x}) d\mathbb{P}_n(\vec{x}) \right)^{1/p} \tag{36}$$

- Based on this relation, WAEs introduce encoders $q_\phi(\vec{z}|\vec{x})$ and target

$$\theta^* \in \arg\min_\theta \left(\inf_{\phi \in \Phi(\mathbb{P}_n)} \int d^p(\vec{x}, G_\theta(\vec{z})) d\mathbb{Q}_\phi(\vec{z}|\vec{x}) d\mathbb{P}_n(\vec{x}) \right)^{1/p} \tag{37}$$

  where $\Phi(\mathbb{P}_n) := \{\phi \mid \int q_\phi(\vec{z}|\vec{x}) d\mathbb{P}_n(\vec{x}) = p(\vec{z})\}$.
- On the RHS, $q_\phi(\vec{z}|\vec{x})$ can be viewed as an encoder. The constraint in the infimum ensures that the marginal distribution of the posterior distributions matches the prior distributions.

p-Wasserstein Distance: Wasserstein Autoencoder

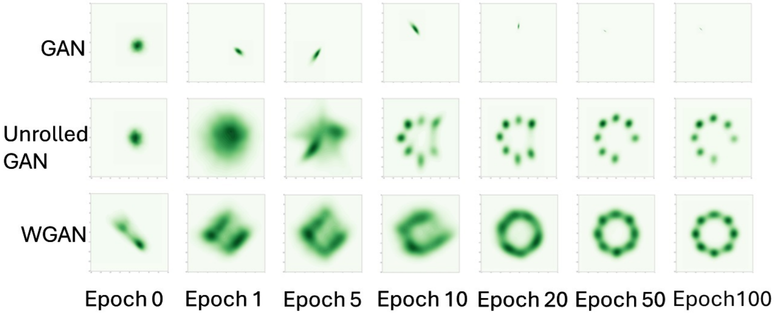
- In implementation, WAEs introduce a penalty term to enforce the constraint on ϕ. The loss can be expressed as:

$$\int d^p(\vec{x}, G_\theta(\vec{z}))d\mathbb{Q}_\phi(\vec{z}|\vec{x})d\mathbb{P}_n(\vec{x}) + \lambda\mathcal{D}_{\vec{Z}}\left(\int q_\phi(\vec{z}|\vec{x})d\mathbb{P}_n(\vec{x}), p(\vec{z})\right) \tag{38}$$

  where $\mathcal{D}_{\vec{Z}}$ indicates the statistical distance applied to the distributions of $\vec{Z}$. WAEs typically use JS divergence and MMD (Maximum Mean Discrepancy) as measures for $\mathcal{D}_{\vec{Z}}$.
- Compared with the loss of VAEs, the negative ELBO, the penalty term changes from matching $q_\phi(\vec{z}|\vec{x})$ directly with $p(\vec{z})$ to matching $\int q_\phi(\vec{z}|\vec{x})d\mathbb{P}_n(\vec{x})$ with $p(\vec{z})$.
- This difference in losses, motivated by theoretical results, may explain why WAEs often yield sharper and more plausible generative results compared to VAEs.

Generation Results: WAEs and WGANs

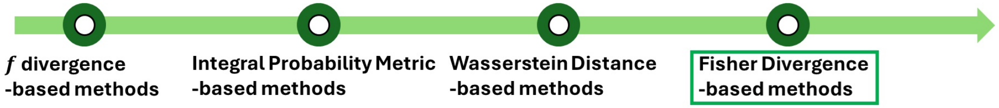
(*a*) Sharp generation results from WAEs



(*b*) Preventing mode collapse with WGANs

Images are edited from Arjovsky et al. (2017) and Tolstikhin et al. (2018).

Emergence of Fisher Divergence-based Methods


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods

- IPM and Wasserstein distance-based methods have alleviated optimization issues; however, adversarial training is still practically difficult.
- Recent works have focused on score functions instead of densities, using estimated scores to generate data.

Recapping Fisher Divergence

- Fisher divergence (Johnson, 2004) is the expected difference between the (Stein) scores (Liu et al., 2016) of two distributions. It can be expressed as:

$$\mathrm{FD}(p_n \parallel p_\theta) = \int \left\| \nabla_{\vec{x}} \log p_n(\vec{x}) - \nabla_{\vec{x}} \log p_\theta(\vec{x}) \right\|^2 p_n(\vec{x})\, d\vec{x}. \tag{39}$$

Fisher Divergence: Score Matching Estimation

1. **Score Matching Estimation**: Score matching estimation (Hyvärinen, 2005) was proposed targeting Fisher divergence in learning distributions.
   - Let $S_\theta(\vec{x}) := \nabla_{\vec{x}} \log p_\theta(\vec{x})$. Then,

$$\mathrm{FD}(p_n || p_\theta) = \int \Big(\mathrm{tr}(\nabla_{\vec{x}} S_\theta(\vec{x})) + \frac{1}{2}\|S_\theta(\vec{x})\|^2\Big) p_n(\vec{x})\, d\vec{x} \tag{40}$$

up to a constant addition and sign-preserving multiplication. We assume that $S_\theta(\vec{x})p_n(\vec{x})$ vanishes at the boundary, e.g., $(x_1, \ldots, x_{i-1}, \pm\infty, x_{i+1}, \ldots, x_m)$.
Proof:

$$\begin{aligned}\mathrm{FD}(p_n || p_\theta) &:= \int \|\nabla_{\vec{x}} \log p_n(\vec{x}) - S_\theta(\vec{x})\|^2 p_n(\vec{x})\, d\vec{x} \\ &= C - 2\int \Big(S_\theta^T(\vec{x}) \nabla_{\vec{x}} \log p_n(\vec{x})\Big) p_n(\vec{x})\, d\vec{x} + \int \|S_\theta(\vec{x})\|^2 p_n(\vec{x})\, d\vec{x}.\end{aligned} \tag{41}$$

Here, $\int \Big(S_\theta^T(\vec{x}) \nabla_{\vec{x}} \log p_n(\vec{x})\Big) p_n(\vec{x})\, d\vec{x}$ equals $-\int \mathrm{tr}(\nabla_{\vec{x}} S_\theta(\vec{x})) p_n(\vec{x})\, d\vec{x}$.

Fisher Divergence: Score Matching Estimation

Proof (Cont.): Let $\vec{X}_{-i} := (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m)^T$. Then,

$$\begin{aligned}\int \Big(S_\theta^T(\vec{x})\nabla_{\vec{x}} \log p_n(\vec{x})\Big) p_n(\vec{x})\, d\vec{x} &= \int S_\theta^T(\vec{x})\nabla_{\vec{x}} p_n(\vec{x})\, d\vec{x} \\ &= \sum_{i=1}^{m} \int \left(\int S_\theta(\vec{x})_i \frac{\partial}{\partial x_i} p_n(\vec{x})\, dx_i \right) d\vec{x}_{-i}.\end{aligned} \tag{42}$$

Since $S_\theta(\vec{x})p_n(\vec{x})$ vanishes at the boundary, by partial integration, we have

$$\int S_\theta(\vec{x})_i \frac{\partial}{\partial x_i} p_n(\vec{x})\, dx_i = -\int \left(\frac{\partial}{\partial x_i} S_\theta(\vec{x})_i\right) p_n(\vec{x})\, dx_i. \tag{43}$$

Thus, $\mathrm{FD}(p_n||p_\theta) = C + \int \Big(2\mathrm{tr}(\nabla_{\vec{x}} S_\theta(\vec{x})) + \|S_\theta(\vec{x})\|^2\Big) p_n(\vec{x}) d\vec{x}$, which concludes the proof.

Fisher Divergence: Sliced Score Matching

2. **Sliced Score Matching**:
   - In the objective of score matching estimation, $\int \left(\mathrm{tr}(\nabla_{\vec{x}} S_\theta(\vec{x})) + \frac{1}{2}\|S_\theta(\vec{x})\|^2\right) p_n(\vec{x})\, d\vec{x}$, the Hessian term poses another computational challenge.
   - Sliced score matching (Song et al., 2020) targets sliced Fisher divergence (SFD),

$$\mathrm{SFD}(p_n||p_\theta) := \int \left\| \vec{v}^T \nabla_{\vec{x}} \log p_n(\vec{x}) - \vec{v}^T \nabla_{\vec{x}} \log p_\theta(\vec{x}) \right\|^2 p_n(\vec{x}) p(\vec{v}) d\vec{x} d\vec{v}, \qquad (44)$$

   to overcome this limitation.
   - The SFD is the average difference between randomly projected scores.

Fisher Divergence: Sliced Score Matching

- In a similar way used in score matching estimation,

$$\mathrm{SFD}(p_n||p_\theta) = \int \left(\vec{v}^T \nabla_{\vec{x}} S_\theta(\vec{x})\vec{v} + \frac{1}{2}(\vec{v}^T S_\theta(\vec{x}))^2\right) p_n(\vec{x}) p(\vec{v}) d\vec{x} d\vec{v} \tag{45}$$

  up to a constant addition and sign-preserving multiplication.
- By changing the target statistical distances from FD to SFD, the computational bottleneck shifts from computing $\mathrm{tr}\left(\nabla_{\vec{x}} S_\theta(\vec{x})\right)$ to computing $\vec{v}^T \nabla_{\vec{x}} S_\theta(\vec{x}) = \nabla_{\vec{x}}\left(\vec{v}^T S_\theta(\vec{x})\right)$, which is numerically less demanding.
- When $p(\vec{v})$ is the multivariate standard Gaussian distribution, the equation $\int \left(\vec{v}^T S_\theta(\vec{x})\right)^2 d\vec{v} = \|S_\theta(\vec{x})\|^2$ holds, further reducing the computational cost.

Fisher Divergence: Noise Conditional Score Network

3. **Noise Conditional Score Network (NCSN)**: NCSNs (Song and Ermon, 2019) are score-based generative models that use estimated scores $S_\theta(\vec{x})$ to generate data.
   - The key idea is to introduce Langevin dynamics in the sampling process. Langevin dynamics describes the stochastic movement of a fluid particle located at $\vec{X}(t)$:
   $$m\frac{d^2\vec{X}(t)}{dt^2} = -\nabla_{\vec{x}=\vec{X}(t)}U(\vec{x}) - \lambda\frac{d\vec{X}(t)}{dt} + \sqrt{2\lambda k_B T}\vec{B}(t), \tag{46}$$
   where m is the mass, U is the potential functions, λ is the damping coefficient, k_B is the Boltzmann constant, T is the temperature, and $\vec{B}(t)$ represents the Brownian motion.
   - In the overdamped case, where the inertial force is negligible, when $\lambda = 1$, we get
   $$d\vec{X}(t) = -\nabla_{\vec{x}=\vec{X}(t)}U(\vec{x})\,dt + \sqrt{2k_B T}\,d\vec{B}(t) \tag{47}$$
   where $d\vec{B}(t) \sim N(0, dtI_m)$.[10] Its stationary distribution is the Boltzmann distribution with energy $U/(k_B T)$, $p(\vec{x}(\infty)) \propto \exp\Big(-U(\vec{x}(\infty))/(k_B T)\Big)$.

[10] This is a special case of the Itô drift-diffusion process.

Fisher Divergence: Noise Conditional Score Network

- By substituting $U(\vec{x}) = -\log p_n(\vec{x})$ and setting $T = 1/k_B$, we obtain:

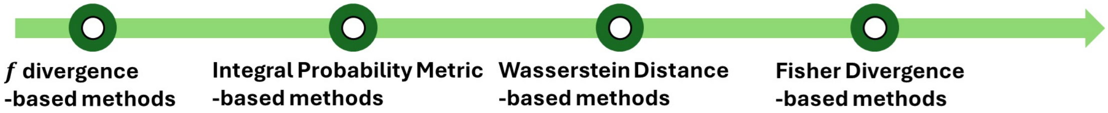
$$d\vec{X}(t) = \nabla_{\vec{x}} \log p_n(\vec{x})\, dt + \sqrt{2dt}\,\vec{\mathcal{E}}(t), \tag{48}$$

  where $\vec{\mathcal{E}}(t) \sim N(0, I_m)$, and the corresponding stationary distribution is $p_n(\vec{x})$.
- The discrete approximation with $dt = \eta/2$ and $S_{\theta^*}(\vec{x})$ results in the following iterative sampling process:

$$\vec{X}(t) = \vec{X}(t-1) + (\eta/2)S_{\theta^*}(\vec{X}(t-1)) + \sqrt{\eta}\vec{\mathcal{E}}(t), \tag{49}$$

  where $\vec{X}(T)$ approximately follows $p_n(\vec{x})$.
- Since the initial points are likely to lie in low-density regions, NCSNs employ the denoising score matching method (Vincent, 2011). They add noise to the data, $\vec{X} + \sigma\vec{\mathcal{E}}$, learn its score $S_\theta(\vec{x}; \sigma)$, and use $S_\theta(\vec{x}; \sigma)$ with a sufficiently small σ for effective sampling.

Summary


f divergence -based methods
Integral Probability Metric -based methods
Wasserstein Distance -based methods
Fisher Divergence -based methods

- We have reviewed recent developments in deep generative models, with a particular focus on targeted statistical distances.
- Advanced topics include:
  1. Introducing new statistical distances,
  2. Theoretical analysis of estimation and approximation errors,
  3. Development of statistical models tailored to specific data structures, such as temporal or multi-modal data.

Thank You

MICHIGAN STATE
UNIVERSITY

References I

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

Bickel, P. J. and Freedman, D. A. (1981). Some asymptotic theory for the bootstrap. *The annals of statistics*, 9(6):1196–1217.

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, 2:1122–1128.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142.

Freedman, D. A. (1981). Bootstrapping regression models. *The annals of statistics*, 9(6):1218–1228.

References II

Georghiades, A. S., Belhumeur, P. N., and Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gross, R., Matthews, I., Cohn, J., Kanade, T., and Baker, S. (2010). Multi-pie. *Image and vision computing*, 28(5):807–813.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, 30.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (2004). *Fundamentals of convex analysis*. Springer Science & Business Media.

References III

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

Johnson, O. (2004). *Information theory and the central limit theorem*. World Scientific.

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422.

Kim, J.-H., Zhang, Y., Han, K., Wen, Z., Choi, M., and Liu, Z. (2021). Representation learning of resting state fmri with variational autoencoder. *NeuroImage*, 241:118423.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

References IV

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017). Mmd gan: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30.

Li, T., Tian, Y., Li, H., Deng, M., and He, K. (2024). Autoregressive image generation without vector quantization. *arXiv preprint arXiv:2406.11838*.

Li, Y., Swersky, K., and Zemel, R. (2015). Generative moment matching networks. In *International conference on machine learning*, pages 1718–1727. PMLR.

Liu, Q., Lee, J., and Jordan, M. (2016). A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR.

Mallows, C. L. (1972). A note on asymptotic joint normality. *The Annals of Mathematical Statistics*, pages 508–515.

References V

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. (2017). Unrolled generative adversarial networks. In *International Conference on Learning Representations*.

Monge, G. (1781). Mémoire sur la théorie des déblais et des remblais. *Mem. Math. Phys. Acad. Royale Sci.*, pages 666–704.

Mroueh, Y., Li, C.-L., Sercu, T., Raj, A., and Cheng, Y. (2017). Sobolev gan. *arXiv preprint arXiv:1711.04894*.

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in applied probability*, 29(2):429–443.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279.

Plaut, E. (2018). From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*.

References VI

Radford, A. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Radford, A. (2018). Improving language understanding by generative pre-training.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr.

Rényi, A. (1961). On measures of entropy and information. In *Proceedings of the fourth Berkeley symposium on mathematical statistics and probability, volume 1: contributions to the theory of statistics*, volume 4, pages 547–562. University of California Press.

Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*, 55(58-63):94.

References VII

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y., Garg, S., Shi, J., and Ermon, S. (2020). Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584. PMLR.

Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. (2003). Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schölkopf, B. (2018). Wasserstein auto-encoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Uppal, A., Singh, S., and Póczos, B. (2019). Nonparametric density estimation & convergence rates for gans under besov ipm losses. *Advances in neural information processing systems*, 32.

References VIII

Villani, C. (2021). *Topics in optimal transportation*, volume 58. American Mathematical Soc.

Villani, C. et al. (2009). *Optimal transport: old and new*, volume 338. Springer.

Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

Wolterink, J. M., Dinkla, A. M., Savenije, M. H., Seevinck, P. R., van den Berg, C. A., and Išgum, I. (2017). Deep mr to ct synthesis using unpaired data. In *International workshop on simulation and synthesis in medical imaging*, pages 14–23. Springer.