Lecture 3: Deep generative models

Minwoo Chae

Department of Industrial and Management Engineering
Pohang University of Science and Technology

KMS-NIMS Summer School on AI, 2025

Outline

Introduction

- Suppose that $\mathbf{X}_1, \ldots, \mathbf{X}_n \overset{\text{iid}}{\sim} P_0$.
  - D-dimensional observations
- The goal is to estimate P_0 or certain functionals of it, such as:
  - The density of P_0
  - The expectation under P_0
  - The support of P_0 (e.g., a manifold)
- Several approaches are available, including:
  - Kernel density estimation
  - Wavelet-based methods
  - Nonparametric Bayesian methods

Deep generative models

- A deep generative model refers to an indirect method for estimating P_0 (or related quantities), which involves:
  1. enabling efficient sampling from an estimated distribution $\hat{P}$ instead of directly estimating P_0,
  2. parameterizing unknown functions using DNNs.
- Recently, deep generative models have achieved remarkable success in modeling high-dimensional data.

Functions parameterized by DNNs

- In deep generative models, the distribution P_0 is parameterized by a function.
- This function is, in turn, parameterized by a DNN.
- Common parameterization approaches include:
  - Generator-based methods
  - Score-based methods
  - Vector fields (defining probability flows)

Generator-based methods

- One can model $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{g}(\mathbf{Z}),$$

  where
  - $\mathbf{Z}$: a d-dimensional latent vector with a known distribution P_Z
  - $\mathbf{g}$: the generator, a map from $\mathbb{R}^d$ to $\mathbb{R}^D$
- Two typical regimes:
  - $d < D$ (low-dimensional latent space)
  - $d = D$ (dimension-preserving mapping)

Generator-based methods (cont.)

- Given a generator $\hat{\mathbf{g}}$, one can generate a sample by
  1. drawing $\mathbf{Z}$ from P_Z, and
  2. computing $\hat{\mathbf{g}}(\mathbf{Z})$.
- The distribution of $\hat{\mathbf{g}}(\mathbf{Z})$ serves as an estimator of P_0.
- Popular approaches for estimating $\hat{\mathbf{g}}$ include:
  - Variational autoencoders (VAEs)
  - Normalizing flows (NFs)
  - Generative adversarial networks (GANs)

Score-based methods

- Suppose that P_0 possesses a Lebesgue density p_0.
- The function $\mathbf{s}_0(\mathbf{x}) = \nabla(\log p_0)(\mathbf{x})$ is called the score function of p_0.
- Score-based methods aim to estimate $\mathbf{s}_0(\mathbf{x})$.
- Once an estimator $\hat{\mathbf{s}}$ is obtained, samples can be generated, for example, via Langevin diffusion.

Score-based methods (cont.)

- Langevin diffusion:

$$\mathrm{d}\mathbf{X}_t = \frac{1}{2}\nabla(\log p_0)(\mathbf{X}_t)\,\mathrm{d}t + \mathrm{d}\mathbf{B}_t,$$
$$\mathrm{d}\hat{\mathbf{X}}_t = \frac{1}{2}\hat{\mathbf{s}}(\hat{\mathbf{X}}_t)\,\mathrm{d}t + \mathrm{d}\mathbf{B}_t,$$

  where $\mathbf{B}_t$ denotes a standard Brownian motion.
- The distribution $\hat{P}$ can be defined as the stationary distribution (limit law) of $\hat{\mathbf{X}}_t$.
- One can discretize the estimated Langevin diffusion as

$$\mathbf{X}_i = \mathbf{X}_{i-1} + \frac{\epsilon}{2}\hat{\mathbf{s}}(\mathbf{X}_{i-1}) + \sqrt{\epsilon}\mathbf{Z}_i,$$

  where $\mathbf{Z}_i \sim \mathcal{N}(\mathbf{0}_D, \mathbb{I}_D)$.

Remark

- Compared to score-based methods, generator-based methods are much easier for sample generation.
- On the other hand, learning the generator function is generally more challenging than estimating the score function.
- We first study the generator-based methods, with a particular focus on VAE and GAN.
- Then, we turn to score-based methods.

Outline

Notations

- Let $P_{\mathbf{g}}$ denote the law of $\mathbf{g}(\mathbf{Z})$ (i.e., the pushforward of P_Z).
- Let $P_{\mathbf{g},\sigma}$ be the law of $\mathbf{g}(\mathbf{Z}) + \boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}_D, \sigma^2 \mathbb{I}_D)$, i.e.,

$$P_{\mathbf{g},\sigma} = P_{\mathbf{g}} * \mathcal{N}(\mathbf{0}_D, \sigma^2 \mathbb{I}_D).$$

- For $\sigma > 0$, $P_{\mathbf{g},\sigma}$ has the density

$$\begin{aligned} p_{\mathbf{g},\sigma}(\mathbf{x}) &= \int \phi_\sigma(\mathbf{x} - \mathbf{z}) \, \mathrm{d}P_{\mathbf{g}}(\mathbf{z}) \\ &= \int \phi_\sigma(\mathbf{x} - \mathbf{g}(\mathbf{z})) \, \mathrm{d}P_Z(\mathbf{z}), \end{aligned}$$

  where ϕ_σ is the density of $\mathcal{N}(\mathbf{0}_D, \sigma^2 \mathbb{I}_D)$.

Variational autoencoders

- For a given class $\mathcal{G}$ of DNN functions, consider the model class

$$\mathcal{P} = \left\{ P_{\mathbf{g},\sigma} : \mathbf{g} \in \mathcal{G},\ \sigma \in [\sigma_{\min}, \sigma_{\max}] \right\},$$

  which is a Gaussian mixture model where the mixing distribution is parametrized by $\mathbf{g}$.

- One may estimate $\mathbf{g}$ via maximum likelihood:

$$(\hat{\mathbf{g}}, \hat{\sigma}) = \underset{(\mathbf{g},\sigma):P_{\mathbf{g},\sigma}\in\mathcal{P}}{\operatorname{argmax}} \sum_{i=1}^{n} \log p_{\mathbf{g},\sigma}(\mathbf{X}_i).$$

Variational autoencoders (cont.)

- A variational autoencoder (VAE) is a specific algorithm for approximating the MLE.
  - It employs a variational inference approach.
- In VAE, the log-likelihood is replaced by the evidence lower bound (ELBO),
  - which is computationally more tractable.
- There exist several other computational methods for approximating the MLE,
  - *e.g.*, expectation–maximization (EM) based methods.

Kingma, D. P. & Welling, M. "Auto-encoding variational Bayes". *Proc. ICLR*. 2014

Rezende, D. J., Mohamed, S. & Wierstra, D. "Stochastic backpropagation and approximate inference in deep generative models". *Proc. ICML*. 2014

Variational autoencoders (cont.)

- Suppose that $\mathbf{g}$ is parameterized by a DNN, and let θ denote the network parameters (possibly including σ).
- For simplicity, let $p_\theta(\cdot) := p_{\mathbf{g},\sigma}(\cdot)$.
- Let $q_\psi(\mathbf{z} \mid \mathbf{x})$ be a conditional density parameterized by ψ, and define

$$\mathcal{L}(\theta, \psi; \mathbf{x}) := \mathbb{E}_{q_\psi(\cdot|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{Z})}{q_\psi(\mathbf{Z} \mid \mathbf{x})} \right) \right],$$

  where $p_\theta(\cdot, \cdot)$ denotes the joint density of $(\mathbf{X}, \mathbf{Z})$.

Variational autoencoders (cont.)

- $\mathcal{L}(\theta, \psi; \mathbf{x})$ is often referred to as the ELBO (evidence lower bound) because

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \mathcal{L}(\theta, \psi; \mathbf{x}) + K\big(q_\psi(\cdot \mid \mathbf{x}),\, p_\theta(\cdot \mid \mathbf{x})\big) \\ &\geq \mathcal{L}(\theta, \psi; \mathbf{x}),\end{aligned}$$

  where $K(\cdot, \cdot)$ denotes the Kullback–Leibler divergence.
- If $q_\psi(\cdot \mid \mathbf{x})$ is sufficiently close to $p_\theta(\cdot \mid \mathbf{x})$, then the ELBO closely approximates $\log p_\theta(\mathbf{x})$.
- If the class $\{q_\psi(\cdot \mid \mathbf{x}) : \psi \in \Psi\}$ is rich enough to approximate $p_\theta(\cdot \mid \mathbf{x})$ for all $\theta \in \Theta$, then maximizing the ELBO yields an approximate MLE.

Variational autoencoders (cont.)

- An approximate MLE can be obtained by

$$(\hat{\theta}, \hat{\psi}) = \operatorname*{argmax}_{(\theta,\psi)\in\Theta\times\Psi} \sum_{i=1}^{n} \mathcal{L}(\theta, \psi; \mathbf{X}_i).$$

- In VAE, the variational distribution $q_\psi(\cdot \mid \mathbf{x})$ is typically modeled as a Gaussian:

$$q_\psi(\mathbf{z} \mid \mathbf{x}) = \phi_{\sigma_\psi(\mathbf{x})}(\mathbf{z} - \mu_\psi(\mathbf{x})),$$

  where ϕ_σ denotes the density of a Gaussian distribution with standard deviation σ.
  - Both μ_ψ and σ_ψ are implemented as DNNs in practice.

Computation of ELBO

- Gradient-based optimization of the ELBO is not straightforward.
- The key step is computing the gradient $\nabla_{(\theta,\psi)}\mathcal{L}(\theta,\psi;\mathbf{x})$.
- Observe that

$$\mathcal{L}(\theta,\psi;\mathbf{x}) = \mathbb{E}_{q_\psi(\cdot|\mathbf{x})} \log p_\theta(\mathbf{x} \mid \cdot) - K\big(q_\psi(\cdot \mid \mathbf{x}), p_Z\big),$$

  where p_Z is the prior density of $\mathbf{Z}$.
- When p_Z is a Gaussian density, the gradient of the KL term with respect to ψ can be computed analytically.

Computation of ELBO (cont.)

- Note that

$$\mathbb{E}_{q_\psi(\cdot|\mathbf{x})} \log p_\theta(\mathbf{x} \mid \cdot) = \mathbb{E}_{\mathbf{Y}\sim\mathcal{N}(\mathbf{0},\mathbb{I})} \log p_\theta(\mathbf{x} \mid \mu_\psi(\mathbf{x}) + \sigma_\psi(\mathbf{x})\mathbf{Y}),$$

  where $\mathbf{Y}$ is a standard Gaussian random variable.
- This reparameterization allows us to approximate the gradient of the expectation via Monte Carlo.

Remark

- The variational posterior $q_{\hat{\psi}}(\cdot \mid \mathbf{x})$ can be used for data compression.
- In practice, the design of the network architecture is as important as the optimization algorithm.
  - This aspect is not covered in this lecture.
- Once an estimator $(\hat{\mathbf{g}}, \hat{\sigma})$ (parametrized by θ in previous slides) is obtained, we define the implicit density and distribution estimators as

$$\hat{p} = p_{\hat{\mathbf{g}},\hat{\sigma}}, \quad \hat{P} = P_{\hat{\mathbf{g}}}.$$

  - There is a minor notational inconsistency in this lecture regarding the use of uppercase vs lowercase letters.

Outline

Normalizing flows

- For diffeomorphic $\mathbf{g}$ with an invertible Jacobian, $P_{\mathbf{g}}$ admits a density:

$$p_{\mathbf{g}}(\mathbf{x}) = p_Z\big(\mathbf{g}^{-1}(\mathbf{x})\big)\det(\nabla \mathbf{g}^{-1}(\mathbf{x})).$$

- For a class $\mathcal{G}$ of such generators, the MLE is defined as

$$\hat{\mathbf{g}} = \operatorname*{argmax}_{\mathbf{g}\in\mathcal{G}} \sum_{i=1}^{n} \log p_{\mathbf{g}}(\mathbf{X}_i).$$

- Normalizing flows (NF) refer to methods for modeling $p_{\mathbf{g}}$ in a computationally tractable way.

Dinh, L., Krueger, D. & Bengio, Y. "NICE: Non-linear independent components estimation". *Proc. ICLR*. 2015

Rezende, D. & Mohamed, S. "Variational inference with normalizing flows". *Proc. ICML*. 2015

Normalizing flows (cont.)

- For MLE to be computationally tractable, it is essential that $\det(\nabla \mathbf{g}^{-1}(\mathbf{x}))$ be easy to compute.
- This can be achieved by composing simple diffeomorphisms.
- If $\mathbf{g} = \mathbf{g}_2 \circ \mathbf{g}_1$ with diffeomorphic $\mathbf{g}_1$ and $\mathbf{g}_2$, then

$$\det(\nabla \mathbf{g}^{-1}(\mathbf{x})) = \det(\nabla \mathbf{g}_1^{-1}(\mathbf{g}_2^{-1}(\mathbf{x})))\det(\nabla \mathbf{g}_2^{-1}(\mathbf{x})).$$

Normalizing flows (cont.)

- Commonly used simple flows include:
  - Radial flows
  - Coupling flows
  - Autoregressive flows
  - Residual flows
- A key advantage of (some) normalizing flows is that they allow explicit evaluation of the density $q_{\mathbf{g}}(\mathbf{x})$.

Outline

Generative adversarial networks

- Let $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$ be an i.i.d. sample from P_Z.
- For a class $\mathcal{F}$ of functions from $\mathbb{R}^D$ to $(0,1)$, generative adversarial networks (GANs) construct $\hat{\mathbf{g}}$ by solving :

$$\underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}} \sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \left\{\log f(\mathbf{X}_i) + \log\left(1 - f(\mathbf{g}(\mathbf{Z}_i))\right)\right\}.$$

- In the GAN framework, $\mathcal{G}$ and $\mathcal{F}$ are referred to as the generator and discriminator classes, respectively.
  - In practice, both are parametrized by DNNs.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. "Generative adversarial nets". *Proc. NIPS*. 2014

Generative adversarial networks (cont.)

- At the population level, GANs aim to solve

$$\underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}} \sup_{f\in\mathcal{F}} \mathbb{E}\left[\log f(\mathbf{X}) + \log\left(1 - f(\mathbf{g}(\mathbf{Z}))\right)\right]$$

$$\Longleftrightarrow \underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}} \sup_{f\in\mathcal{F}} \left\{\int \log f \, \mathrm{d}P_0 + \int \log(1-f) \, \mathrm{d}P_{\mathbf{g}}\right\}$$

- This objective is closely related to minimizing the Jensen–Shannon divergence:

$$\underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}}\, \mathrm{JS}(P_0, P_{\mathbf{g}}),$$

where

$$\mathrm{JS}(P,Q) = \frac{1}{2}\left\{K(P,(P+Q)/2) + K(Q,(P+Q)/2)\right\}.$$

Variations of GANs

- The adversarial training framework of GANs motivates several variations, which can be interpreted as solving the following problem at the population level:

$$\underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}}\, d(P_0, P_{\mathbf{g}}),$$

where d is a discrepancy measure of the form

$$d(P, Q) = \sup_{f\in\mathcal{F}} \left\{ \int f \,\mathrm{d}P - \int h(f) \,\mathrm{d}Q \right\},$$

with a fixed function $h : \mathbb{R} \to \mathbb{R}$.

f-GAN

- For a convex function ϕ, the f-divergence (also known as the Csiszár divergence or Ali–Silvey distance) is defined as

$$D_\phi(P, Q) = \int \phi\left(\frac{\mathrm{d}P}{\mathrm{d}Q}\right) \mathrm{d}Q.$$

- Examples:
  - Kullback–Leibler (KL) divergence (both directions)
  - Total variation distance
  - α-divergence (e.g., squared Hellinger, Pearson χ^2)
  - Jensen–Shannon (JS) divergence
- At the population level, f-GAN aims to solve

$$\underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}}\, D_\phi(P_0, P_\mathbf{g}).$$

f-GAN (cont.)

- It is well known that for any function class $\mathcal{F}$,

$$D_\phi(P, Q) \geq \sup_{f \in \mathcal{F}} \left\{ \int f \, \mathrm{d}P - \int \phi^*(f) \, \mathrm{d}Q \right\},$$

  where ϕ^* is the convex conjugate of ϕ, defined as

$$\phi^*(t) = \sup_{u \in \mathrm{dom}(\phi)} \{ut - \phi(u)\}.$$

- Equality holds if and only if $\partial\phi(p/q) \cap \mathcal{F} \neq \emptyset$.

Nguyen, X., Wainwright, M. J. & Jordan, M. I. "Estimating divergence functionals and the likelihood ratio by convex risk minimization". *IEEE Trans. Inform. Theory*. 2010

f-GAN (cont.)

- The variational representation of f-divergence motivates the f-GAN objective:

$$\underset{\mathbf{g}\in\mathcal{G}}{\text{minimize}} \sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\Big\{f(\mathbf{X}_i) - \phi^*\big(f(\mathbf{g}(\mathbf{Z}_i))\big)\Big\}.$$

- In practice, both $\mathcal{G}$ and $\mathcal{F}$ are parametrized by DNNs.

Nowozin, S., Cseke, B. & Tomioka, R. "f-GAN: Training generative neural samplers using variational divergence minimization". *Proc. NIPS*. 2016

IPM GAN

- For a class $\mathcal{F}$ of real-valued functions on $\mathbb{R}^D$, the $\mathcal{F}$-IPM (Integral Probability Metric) is defined as

$$d_{\mathcal{F}}(P, Q) = \sup_{f \in \mathcal{F}} \left| \int f \, \mathrm{d}P - \int f \, \mathrm{d}Q \right|.$$

- IPM GAN aims to solve

$$\underset{\mathbf{g} \in \mathcal{G}}{\text{minimize}} \, d_{\mathcal{F}}(P_0, P_{\mathbf{g}}).$$

- This leads to the following empirical objective:

$$\underset{\mathbf{g} \in \mathcal{G}}{\text{minimize}} \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left\{ f(\mathbf{X}_i) - f(\mathbf{g}(\mathbf{Z}_i)) \right\}.$$

Müller, A. "Integral probability metrics and their generating classes of functions". *Adv. in Appl. Probab.* 1997

IPM GAN (cont.)

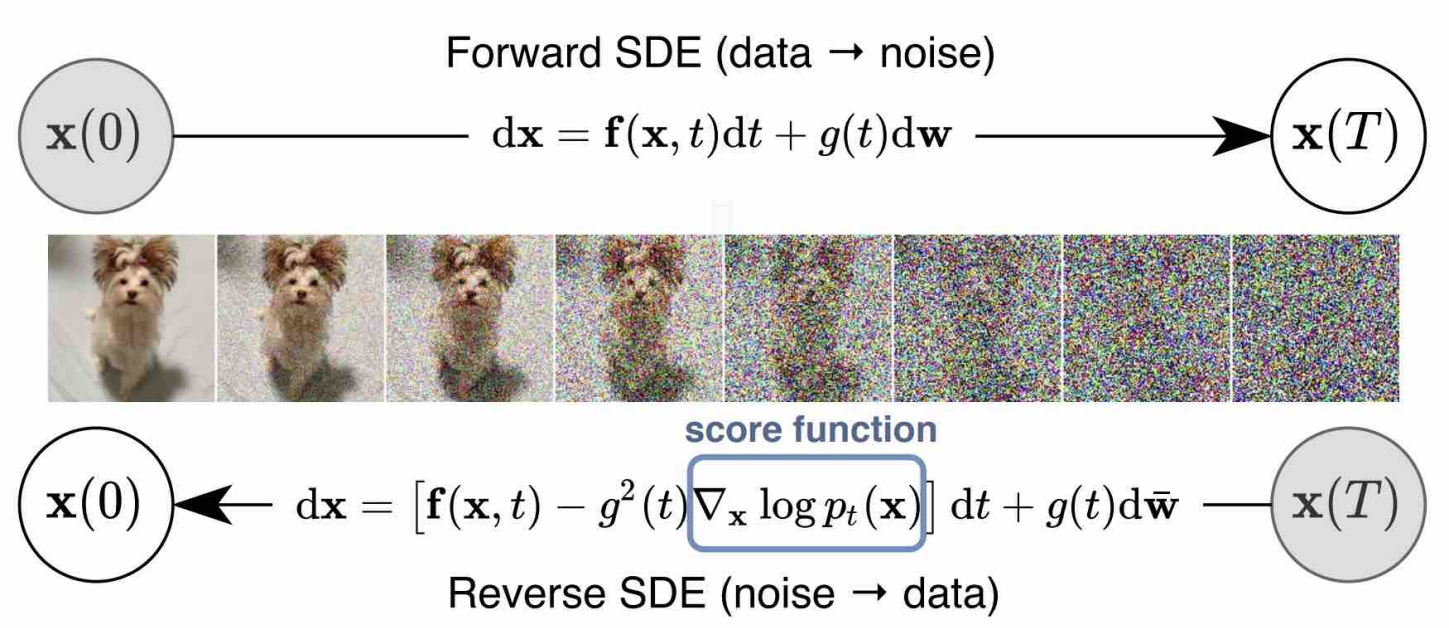
- Example 1: Wasserstein GAN (WGAN)
  - $\mathcal{F} = \{f : \mathrm{Lip}(f) \leq 1\}$ (1-Lipschitz functions).
- Example 2: Maximum Mean Discrepancy (MMD) GAN
  - $\mathcal{F}$ is the unit ball of a reproducing kernel Hilbert space (RKHS).
- In practice, both $\mathcal{G}$ and $\mathcal{F}$ are parametrized by DNNs. However, different choices of $\mathcal{F}$ result in different optimization algorithms.
- A key advantage: IPMs are well-defined even when probability densities do not exist.

Arjovsky, M., Chintala, S. & Bottou, L. "Wasserstein generative adversarial networks". *Proc. ICML*. 2017

Dziugaite, G. K., Roy, D. M. & Ghahramani, Z. "Training generative neural networks via maximum mean discrepancy optimization". *Proc. Conference on Uncertainty in Artificial Intelligence*. 2015

Outline

Introduction



Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S. & Poole, B. "Score-based generative modeling through stochastic differential equations". *Proc. ICLR*. 2021

Score matching

- The estimation of the score function dates back to Hyvärinen (2005).
- For a function $\mathbf{s} : \mathbb{R}^D \to \mathbb{R}^D$, under mild regularity conditions,

$$\frac{1}{2}\mathbb{E}\left[\|\mathbf{s}(\mathbf{X}) - \mathbf{s}_0(\mathbf{X})\|_2^2\right] = \mathbb{E}\left[\operatorname{tr}\left(\nabla \mathbf{s}(\mathbf{X})\right) + \frac{1}{2}\|\mathbf{s}(\mathbf{X})\|_2^2\right] - \frac{1}{2}\mathbb{E}\|\mathbf{s}_0(\mathbf{X})\|_2^2.$$

Hyvärinen, A. "Estimation of non-normalized statistical models by score matching". *J. Mach. Learn. Res.* 2005

Score matching (cont.)

- For $\mathbf{s} : \mathbb{R}^D \to \mathbb{R}^D$, the score matching loss is defined by

$$\ell_{\mathbf{s}}(\mathbf{x}) = \mathrm{tr}\big(\nabla \mathbf{s}(\mathbf{x})\big) + \frac{1}{2}\|\mathbf{s}(\mathbf{x})\|_2^2,$$

  which leads to an M-estimator (or ERM):

$$\begin{aligned}\hat{\mathbf{s}} &= \operatorname*{argmin}_{\mathbf{s}\in\mathcal{S}} \int \ell_{\mathbf{s}}(\mathbf{x})\, \mathbb{P}_n(\mathrm{d}\mathbf{x}) \\ &= \operatorname*{argmin}_{\mathbf{s}\in\mathcal{S}} \left[\frac{1}{n}\sum_{i=1}^{n} \ell_{\mathbf{s}}(\mathbf{X}_i)\right],\end{aligned}$$

  where $\mathcal{S}$ denotes a class of DNNs.

Score matching (cont.)

- The computation of $\text{tr}(\nabla \mathbf{s}(\mathbf{x}))$ is not scalable for large D.
- Several scalable alternatives have been proposed:
  - Sliced score matching (Song et al., 2020)
  - Denoising score matching (Vincent, 2011)

Song, Y., Garg, S., Shi, J. & Ermon, S. "Sliced score matching: A scalable approach to density and score estimation". *Uncertainty in Artificial Intelligence*. 2020

Vincent, P. "A connection between score matching and denoising autoencoders". *Neural Comput.* 2011

Remarks

- In real-world applications, noise is often injected at multiple levels, and the score functions of the resulting noise-perturbed distributions are jointly modeled and estimated.
- A variety of approaches have been proposed, many of which can be understood within a unified framework presented by Song et al. (2021).

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S. & Poole, B. “Score-based generative modeling through stochastic differential equations”. *Proc. ICLR*. 2021

Score-based methods through SDE

- Consider the Ornstein–Uhlenbeck (OU) process

$$\mathrm{d}\mathbf{X}_t = -\frac{1}{2}\mathbf{X}_t \mathrm{d}t + \mathrm{d}\mathbf{B}_t, \quad t \geq 0,$$

  with initial distribution $\mathbf{X}_0 \sim P_0$.

- For the OU process, the transition kernels are explicitly given by

$$\mathbf{X}_{t+s} \mid \mathbf{X}_s = \mathbf{x} \sim \mathcal{N}\left(\mu_t \mathbf{x}, \sigma_t^2 \mathbb{I}_D\right),$$

  where $\mu_t = e^{-t/2}$ and $\sigma_t^2 = 1 - \mu_t^2$.

Score-based methods through SDE (cont.)

- Let $p_t(\cdot \mid \cdot)$ be the transition kernel density of a diffusion process, and define

$$p_t(\mathbf{x}) = \int p_t(\mathbf{x} \mid \mathbf{X}_0 = \mathbf{x}_0) \mathrm{d}P_0(\mathbf{x}_0)$$

  as the marginal density of $\mathbf{X}_t$.
- p_t converges rapidly to the standard Gaussian density as $t \to \infty$.
- With slight abuse of notation, define the score function of p_t by

$$\mathbf{s}_0(t, \mathbf{x}) = \nabla(\log p_t)(\mathbf{x}).$$

Score-based methods through SDE (cont.)

- For a large enough $T > 0$, define the reverse process $\mathbf{Y}_t := \mathbf{X}_{T-t}$.
- Under mild regularity conditions, the reverse process satisfies the SDE (Anderson, 1982)

$$\mathrm{d}\mathbf{Y}_t = \frac{1}{2}\mathbf{Y}_t \mathrm{d}t + \mathbf{s}_0(T-t, \mathbf{Y}_t)\mathrm{d}t + \mathrm{d}\widetilde{\mathbf{B}}_t.$$

- Once an estimator $\hat{\mathbf{s}}(\cdot,\cdot)$ for $\mathbf{s}_0(\cdot,\cdot)$ is obtained, one can simulate the reverse process starting from a standard Gaussian sample.

Score-based methods through SDE (cont.)

- The function $\mathbf{s}_0(\cdot,\cdot)$ can be estimated via score matching.
- Alternatively, one may estimate the conditional expectation

$$\mathbf{m}_0(t,\mathbf{x}) = \mathbb{E}[\mathbf{X}_0 \mid \mathbf{X}_t = \mathbf{x}]$$

  using (weighted) least squares regression.
- These two approaches are equivalent under reparametrization, since

$$\mathbf{s}_0(t,\mathbf{x}) = -\frac{\mathbf{x} - \mu_t \mathbf{m}_0(t,\mathbf{x})}{\sigma_t^2}.$$

Score-based methods through SDE (cont.)

- For any weight function $\lambda(\cdot)$,

$$\mathbf{m}_0 = \underset{\mathbf{m}}{\operatorname{argmin}}\, \mathbb{E}\left[\int_0^T \lambda(t)\|\mathbf{X}_0 - \mathbf{m}(t, \mathbf{X}_t)\|^2 \mathrm{d}t\right].$$

- By modeling $\mathbf{m}$ using a DNN, an estimator $\hat{\mathbf{m}}$ can be constructed via empirical risk minimization.
- Let $\hat{\mathbf{s}}(\cdot, \cdot)$ be the corresponding estimator of the score function.

Score-based methods through SDE (cont.)

- Recall that the reverse process $\mathbf{Y}_t = \mathbf{X}_{T-t}$ satisfies

$$\mathrm{d}\mathbf{Y}_t = \frac{1}{2}\mathbf{Y}_t \mathrm{d}t + \mathbf{s}_0(T-t, \mathbf{Y}_t)\mathrm{d}t + \mathrm{d}\widetilde{\mathbf{B}}_t, \quad \mathbf{Y}_0 \sim P_T.$$

- Define a process $(\hat{\mathbf{Y}}_t)$ using the estimated score $\hat{\mathbf{s}}$ as

$$\mathrm{d}\hat{\mathbf{Y}}_t = \frac{1}{2}\hat{\mathbf{Y}}_t \mathrm{d}t + \hat{\mathbf{s}}(T-t, \hat{\mathbf{Y}}_t)\mathrm{d}t + \mathrm{d}\widetilde{\mathbf{B}}_t, \quad \hat{\mathbf{Y}}_0 \sim \mathcal{N}(\mathbf{0}_D, \mathbb{I}_D).$$

- Define $\hat{P}$ (or $\hat{p}$) as the distribution (or density) of $\hat{\mathbf{Y}}_T$.
  - Algorithmic issues are not discussed in this talk.

Remarks

- Score-based generative models represent the current state of the art in generative modeling.
- Various approaches are available for estimating the score function.
- A key practical concern is the computational cost of sample generation.

Thank you for attention!