Deep Generative Models

Score-Based Generative Models

Hamid Beigy

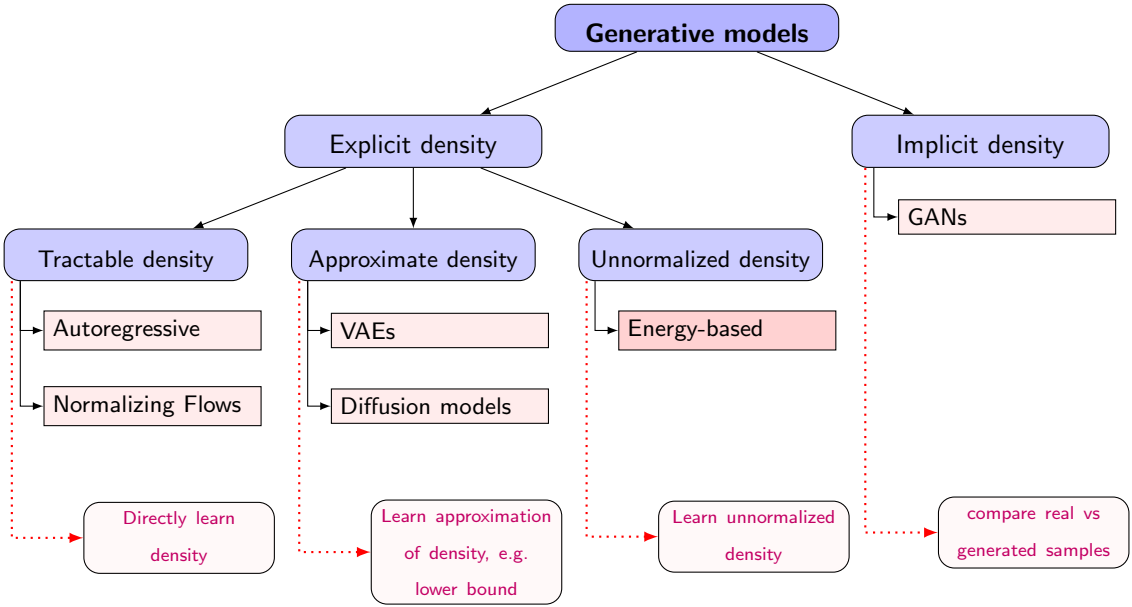
Sharif University of Technology

May 4, 2025

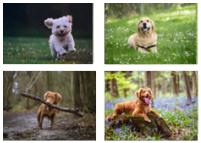
Table of contents

Introduction

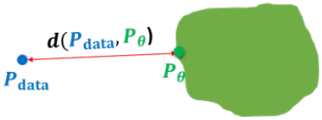
Generative models categories



1. Assume that the observed variable $\mathbf{x}$ is a random sample from an underlying process, whose true distribution $p_{\text{d}}(\mathbf{x})$ is unknown.



$\mathbf{x}_i \sim P_{\text{data}}$
$i = 1, 2, \ldots, n$



2. We attempt to approximate this process with a chosen model, $p_\theta(\mathbf{x})$, with parameters θ such that $\mathbf{x} \sim p_\theta(\mathbf{x})$.

3. Learning is the process of searching for the parameter θ such that $p_\theta(\mathbf{x})$ well approximates $p_{\text{d}}(\mathbf{x})$ for any observed $\mathbf{x}$, i.e.

$$p_\theta(\mathbf{x}) \approx p_{\text{d}}(\mathbf{x})$$

4. We wish $p_\theta(\mathbf{x})$ to be sufficiently flexible to be able to adapt to the data for obtaining sufficiently accurate model and to be able to incorporate prior knowledge.

Credit: Aditya Grover

Autoregressive models

1. Tractable density
2. Density is estimated as

$$p_\theta(\mathbf{x}) = \prod_{j=1}^{m} p_\theta(\mathbf{x}_j \mid \mathbf{x}_{<j})$$

3. Tractable likelihood
4. No inferred latent factors

Normalizing flow models

1. Exact density
2. Density is estimated as

$$p_\theta(\mathbf{x}) = p_\mathbf{z}(\mathbf{z})|\det(\mathbf{J}_f)|$$
$$\text{where } \mathbf{z} = f(\mathbf{x})$$

3. Tractable likelihood
4. Latent feature representation

Latent variable models

1. Approximated density
2. Density is estimated as

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

3. Intractable likelihood
4. Latent feature representation

Generative adversarial networks

1. Implicit density
2. Can optimize f-divergences and Wasserstein distance

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_d(\mathbf{x})}[\log D(x)] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

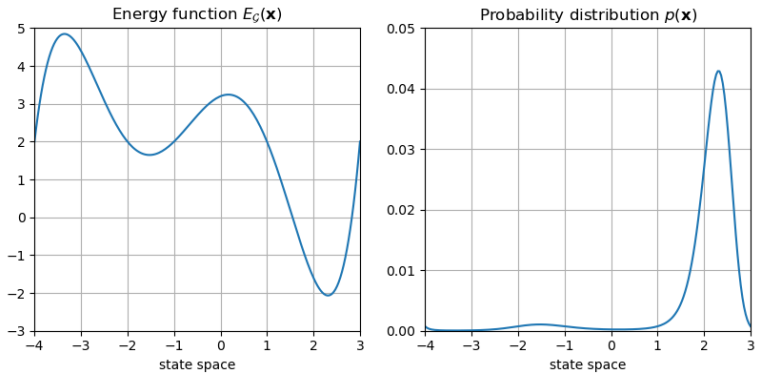
3. Latent feature representation
4. Very flexible model architectures, unstable training, hard evaluation, mode collapse

Representing probability distributions

1. The parametrized versions of the probability density functions

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \exp(-E_\theta(\mathbf{x})) \qquad \text{where} \quad Z_\theta = \int \exp(-E_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}$$

2. A benefit of EBM is that
   - energy functions are not constrained to be non-negative.
   - energy functions can be very flexible parametrized.
3. An energy function and its corresponding probability distributions



Representing probability distributions

1. The density function given by an EBM is

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

2. Evaluation and differentiation of $\log\ p_\theta(\mathbf{x})$ w.r.t. its parameters involves a typically intractable integral.

$$\max_\theta \sum_{k=1}^{m} \log\ p_\theta(\mathbf{x}_k)$$

3. **Pros:**
   - **Extreme flexibility:** can use any function $E_\theta(\mathbf{x})$ you want
4. **Cons:**
   - Sampling from $p_\theta(\mathbf{x})$ is hard.
   - Evaluating and optimizing likelihood $p_\theta(\mathbf{x})$ is hard (learning is hard).
   - No feature learning (but we can add latent variables)
5. **Problem:** A fundamental problem is that computing Z_θ numerically scales exponentially in the number of dimensions of $\mathbf{x}$.

Tractability/flexibility of generative models

1. In generative modeling there are two opposing forces: **tractability** and **flexibility**.
2. Tractable models are usually analytically computable, thus easy to evaluate and fit.
3. But they are usually **not flexible enough to learn the true data structure**.
4. Flexible models can fit arbitrary structures in data.
5. But they are usually **expensive to evaluate, fit, or sample from**
6. **Diffusion/score-matching models** are both **tractable and flexible**.

Representing probability distributions

1. GAN-like quality and better, while having the advantages of explicit probabilistic models.
   - Explicit likelihood computation
   - Representation learning
2. State-of-the results in generation, audio synthesis, shape generation. etc.

3. Score-based models we do not need a tractable normalizing constant.

Score-Based Generative Models

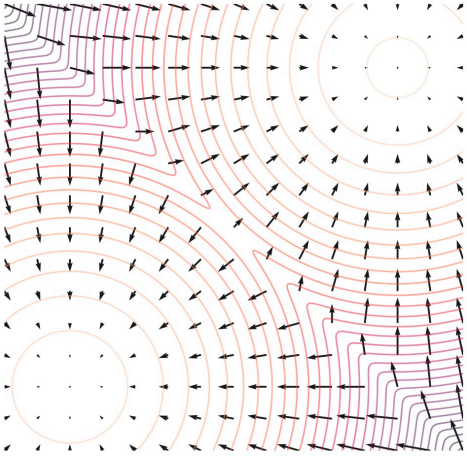
Score function

1. The (Stein) **score function** is the **gradient of the log-probability of a distribution w.r.t. to the input**.

$$\mathbf{s}(\mathbf{x}) = \nabla_{\mathbf{x}} \log\, p(\mathbf{x})$$

2. A model $\mathbf{s}_\theta(\mathbf{x})$, which models the score function explicitly, is a score-based model

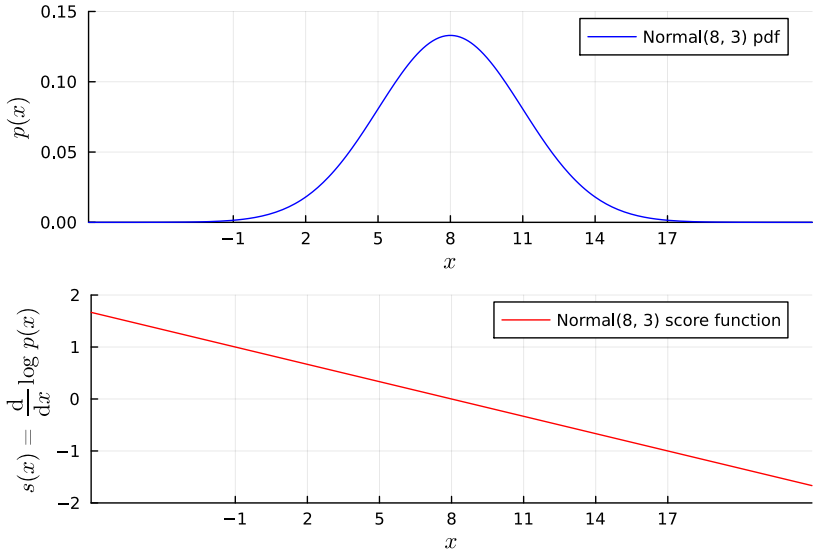
$$\mathbf{s}_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})$$



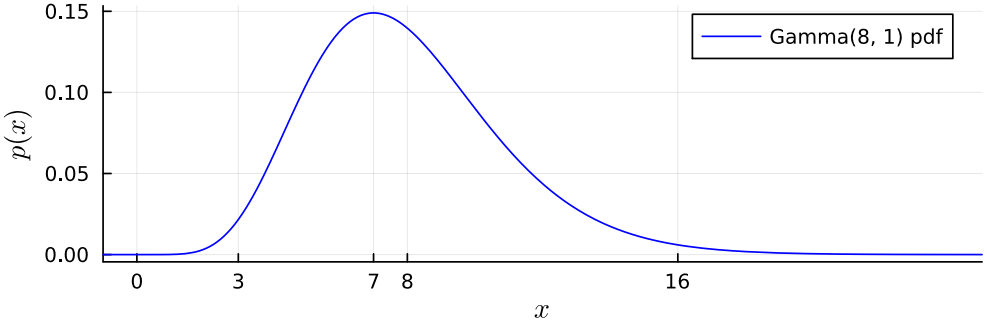
The score function of a mixture of two Gaussians

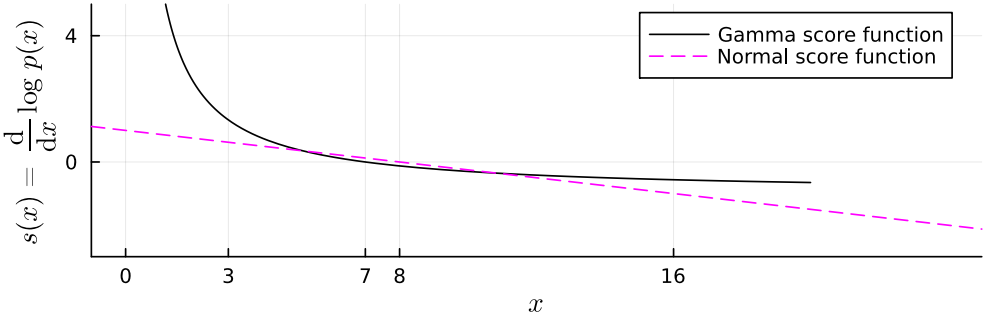
Score function (One-dimensional Gaussian distribution)

1. Consider one-dimensional Gaussian distribution $\mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$.
2. Its log probability $\log \mathcal{N}(\mu, \sigma^2) = -\log\left(\sqrt{2\pi}\ \sigma\right) - \frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2$.
3. The score of point x is $\mathbf{s}(x) = \frac{d \log \mathcal{N}(\mu, \sigma^2)}{dx} = \left(\frac{\mu - x}{\sigma^2}\right)$.

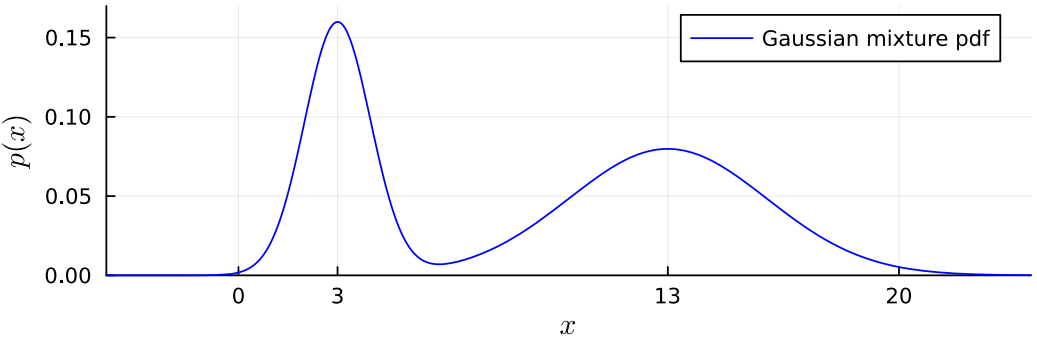


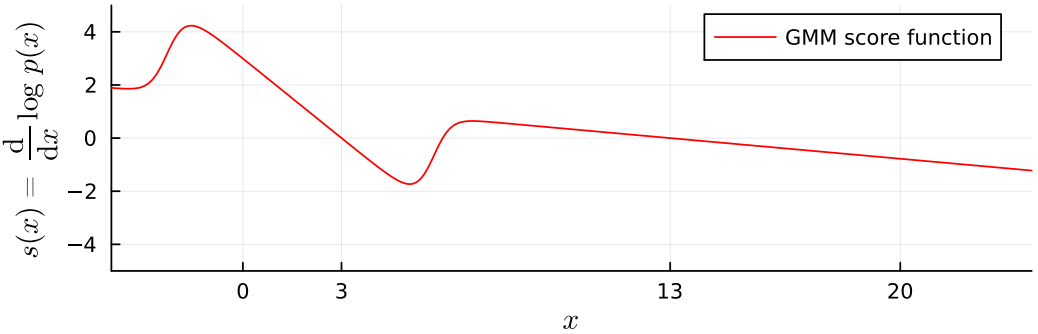
Score function (One-dimensional Gamma distribution)





Score function (One-dimensional GMM distribution)

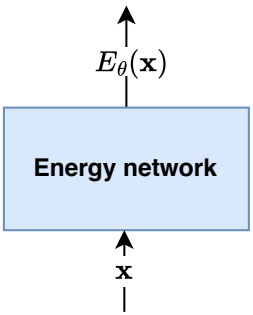




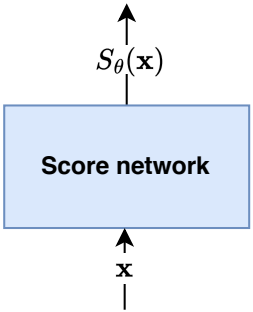
1. Score-based models we do not need a tractable normalizing constant

$$\begin{aligned}\mathbf{s}_\theta(\mathbf{x}) &= \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) \\ &= -\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_\theta}_{=0} = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})\end{aligned}$$

Energy-based model

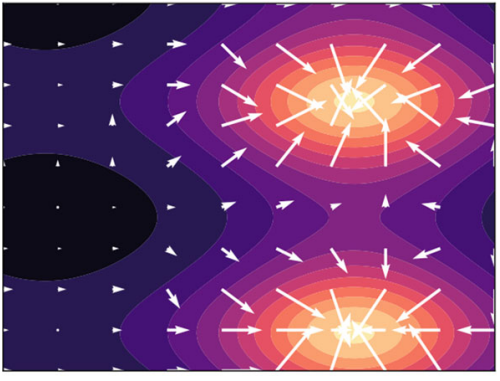


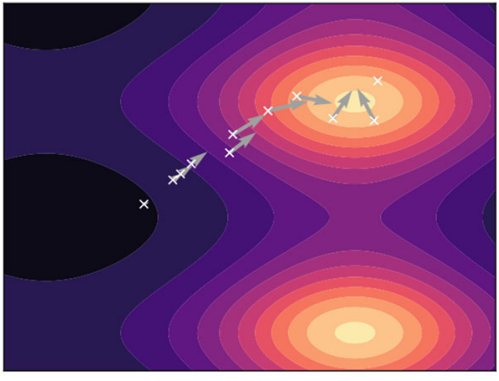
Score-based model



1. In Langevin dynamics, initially draws a sample $\mathbf{x}_0$ from a simple prior distribution.
2. Then uses a process for K steps with step size $\epsilon > 0$ and $\mathbf{z}^k \sim \mathcal{N}(0, \mathbf{I})$:

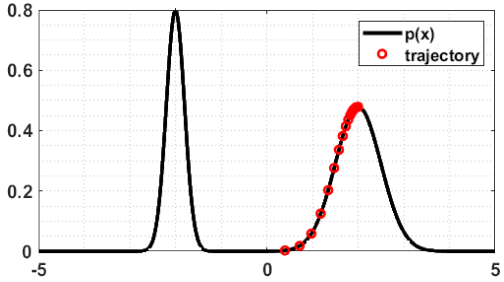
$$\begin{aligned}\mathbf{x}^{k+1} &\leftarrow \mathbf{x}^k + \frac{\epsilon^2}{2} \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) + \epsilon \mathbf{z}^k \\ &= \mathbf{x}^k + \frac{\epsilon^2}{2} \mathbf{s}_\theta(\mathbf{x}) + \epsilon \mathbf{z}^k.\end{aligned}$$



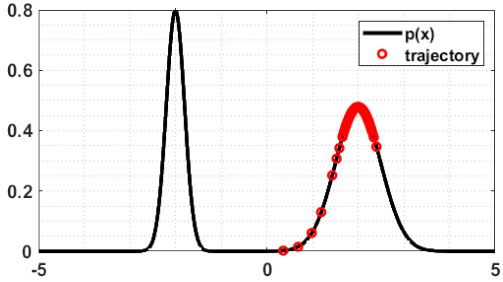


1. In Langevin dynamics, initially draws a sample $\mathbf{x}_0$ from a simple prior distribution.
2. Then uses a process for K steps with step size $\epsilon > 0$ and $\mathbf{z}^k \sim \mathcal{N}(0, \mathbf{I})$:

$$\begin{aligned}\mathbf{x}^{k+1} &\leftarrow \mathbf{x}^k + \frac{\epsilon^2}{2} \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) + \epsilon \mathbf{z}^k \\ &= \mathbf{x}^k + \frac{\epsilon^2}{2} \mathbf{s}_\theta(\mathbf{x}) + \epsilon \mathbf{z}^k.\end{aligned}$$



$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla_{\mathbf{x}} \log p(\mathbf{x}_t)$$



$$\mathbf{x}_{t+1} = \mathbf{x}_t + \tau \nabla_{\mathbf{x}} \log p(\mathbf{x}_t) + \sqrt{2\tau}\mathbf{z}$$

Example 3.3. Following the previous example we again consider a Gaussian mixture

$$p(x) = \pi_1 \mathcal{N}(x \mid \mu_1, \sigma_1^2) + \pi_2 \mathcal{N}(x \mid \mu_2, \sigma_2^2).$$

We choose $\pi_1 = 0.6$. $\mu_1 = 2$, $\sigma_1 = 0.5$, $\pi_2 = 0.4$, $\mu_2 = -2$, $\sigma_2 = 0.2$. Suppose we initialize $M = 10000$ uniformly distributed samples $x_0 \sim \text{Uniform}[-3, 3]$. We run Langevin updates for $t = 100$ steps. The histograms of generated samples are shown in the figures below.

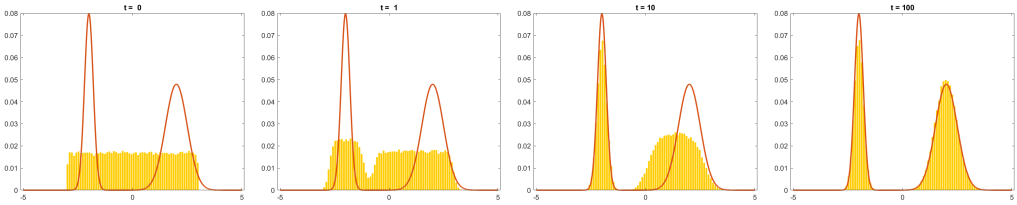


Figure 3.4: Samples generated by Langevin dynamics. Initially the samples are uniformly distributed. As time progresses, the distribution of the samples become the desired distribution.

Score Matching (SM)

1. Let $f(\mathbf{x})$ and $g(\mathbf{x})$ be two continuously differentiable real-valued functions.
2. If $f(\mathbf{x})$ and $g(\mathbf{x})$ have equal first derivatives everywhere, then $f(\mathbf{x}) = g(\mathbf{x}) + \textit{Constant}$.
3. When $f(\mathbf{x})$ and $g(\mathbf{x})$ are log-pdfs with equal first derivatives, the normalization requirement implies that

$$\int \exp(f(\mathbf{x}))d\mathbf{x} = \int \exp(g(\mathbf{x}))d\mathbf{x} = 1$$

and

$$f(\mathbf{x}) \equiv g(\mathbf{x})$$

.

4. We can approximately learn an EBM by matching the first derivatives of its log-pdf to the first derivatives of the log-pdf of the data distribution.
5. If they match, then the EBM captures the data distribution exactly.
6. The **first-order gradient function of a log-pdf** is also called **the score of that distribution** (Hyvärinen 2005).
7. For training EBMs, it is useful to transform the equivalence of distributions to the equivalence of scores, because the score of an EBM can be easily obtained.

1. The main problem is that the probability distribution function $p_{\mathrm{d}}(\mathbf{x})$ is unknown.
2. A simple way to approximate $p_{\mathrm{d}}(\mathbf{x})$ is to use kernel density estimator, denoted by $q(\mathbf{x})$.

$$q(\mathbf{x}) = \frac{1}{m}\sum_{k=1}^{m}\frac{1}{h}K\left(\frac{\mathbf{x}-\mathbf{x}_k}{h}\right),$$

   where h is a hyper-parameter for kernel $K(.)$ and $\mathbf{x}_k$ is kth sample in the training set.

3. By using the definition of kernel density estimator, loss function equals to

$$\begin{aligned}\mathcal{J}_{esm}(\theta) &= \int_{\mathbf{x}} \|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log\, q(\mathbf{x})\|^2\, q(\mathbf{x})\mathrm{d}\mathbf{x} \\ &= \int_{\mathbf{x}} \|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log\, q(\mathbf{x})\|^2 \frac{1}{m}\sum_{k=1}^{m}\frac{1}{h}K\left(\frac{\mathbf{x}-\mathbf{x}_k}{h}\right)\mathrm{d}\mathbf{x} \\ &= \frac{1}{m\times h}\sum_{k=1}^{m}\int_{\mathbf{x}} \|\mathbf{s}_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log\, q(\mathbf{x})\|^2 K\left(\frac{\mathbf{x}-\mathbf{x}_k}{h}\right)\mathrm{d}\mathbf{x}\end{aligned}$$

4. Explicit score matching has a drawback because kernel density estimation is not a very effective way to estimate the actual data distribution when we have a small number of samples in a high-dimensional space.

Score Matching (SM)

1. The score of an EBM can be easily obtained by $\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$.
2. The score does not involve the typically intractable normalizing constant Z_θ.
3. The basic score matching minimizes a discrepancy between two distributions called the **Fisher divergence**:

$$\mathrm{D}_{FS}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x})) = \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log\, p_{\mathrm{d}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})\|^2\right]$$

4. The first term admits a trivial unbiased Monte Carlo estimator using the empirical mean of samples $\mathbf{x} \sim p_{\mathrm{d}}(\mathbf{x})$.
5. The second term is generally impractical to calculate since it requires knowing $\nabla_{\mathbf{x}} \log\, p_{\mathrm{d}}(\mathbf{x})$.

1. Score matching eliminates the data score using integration by parts. To simplify discussion, we consider the Fisher divergence between distributions of 1-D random variables as

$$\begin{aligned}\frac{1}{2}\mathbb{E}_{p_{\mathrm{d}}(x)}\left[\left(\nabla_x \log\, p_{\mathrm{d}}(x) - \nabla_x \log\, p_\theta(x)\right)^2\right] =& \frac{1}{2}\int p_{\mathrm{d}}(x)\left(\nabla_x \log\, p_{\mathrm{d}}(x) - \nabla_x \log\, p_\theta(x)\right)^2 dx \\ =& \underbrace{\frac{1}{2}\int p_{\mathrm{d}}(x)\left(\nabla_x \log\, p_{\mathrm{d}}(x)\right)^2 dx}_{\text{Constant}} \\ &+ \frac{1}{2}\int p_{\mathrm{d}}(x)\left(\nabla_x \log\, p_\theta(x)\right)^2 dx \\ &- \int p_{\mathrm{d}}(x)\nabla_x \log\, p_\theta(x)\nabla_x \log\, p_{\mathrm{d}}(x) dx.\end{aligned}$$

2. By integration by parts, we have

$$\begin{aligned}-\int p_{\mathrm{d}}(x)\nabla_x \log\, p_\theta(x)\nabla_x \log\, p_{\mathrm{d}}(x) dx &= -\int \nabla_x \log\, p_\theta(x)\nabla_x\, p_{\mathrm{d}}(x) dx \\ &= -\, p_{\mathrm{d}}(x)\nabla_x \log\, p_\theta(x)\Big|_{-\infty}^{\infty} \\ &\quad + \int p_{\mathrm{d}}(x)\nabla_x^2 \log\, p_\theta(x) dx \\ &\overset{(i)}{=} \mathbb{E}_{p_{\mathrm{d}}(x)}[\nabla_2^x \log\, p_\theta(x)],\end{aligned}$$

1. The line (i) holds if we assume $p_{\text{d}}(x) \to 0$ as $|x| \to \infty$.
2. Substituting the results of integration by parts into the 1-D Fisher divergence, we obtain

$$\frac{1}{2}\mathbb{E}_{p_{\text{d}}(x)}\Big[\big(\nabla_x \log\, p_{\text{d}}(x) - \nabla_x \log\, p_\theta(x)\big)^2\Big] = \mathbb{E}_{p_{\text{d}}(x)}\big[\nabla_x^2 \log\, p_\theta(x)\big] + \frac{1}{2}\mathbb{E}_{p_{\text{d}}(x)}\Big[\big(\nabla_x \log\, p_\theta(x)\big)^2\Big] + \text{Constant}.$$

   Therefore, the equivalent form of 1-D Fisher divergence does not involve $\nabla_x \log\, p_{\text{d}}(x)$.
3. Generalizing the integration by parts argument to muti-dimensional data, we have the following objective equivalent to Fisher divergenceẽparenciteHyvarinen05.

$$\mathbb{E}_{p_{\text{d}}(\mathbf{x})}\left[\text{tr}\big(\nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\big) + \frac{1}{2}\|\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})\|_2^2\right] + \text{Constant},$$

   where $\nabla_{\mathbf{x}}^2$ denotes the Hessian with respect to $\mathbf{x}$.
4. This objective is known as the implicit score matching objective, because it only involves functions of $\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})$ and it does not depend on the intractable partition function.
5. Therefore, **it is ideal for learning unnormalized probability models**.

1. The Fisher divergence can be rewritten as:

$$\begin{aligned} \mathrm{D}_{FS}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x})) &= \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\left[\mathrm{tr}\big(\nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\big) + \frac{1}{2}\|\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})\|_2^2 \right] + \text{Constant}, \\ &= \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\left[\frac{1}{2}\sum_{i=1}^{D}\left(\frac{\partial E_\theta(\mathbf{x})}{\partial x_i}\right)^2 + \left(\frac{\partial^2 E_\theta(\mathbf{x})}{\partial x_i^2}\right)^2 \right] + \text{Constant} \end{aligned}$$

2. SM only requires the trace of the Hessian, but it is still expensive to compute even with modern hardware and automatic differentiation packages (Martens, Sutskever, and Swersky 2012).
3. For this reason, the implicit SM formulation has only been applied to relatively simple energy functions where computation of the second derivatives is tractable.
4. Score Matching assumes a continuous data distribution with positive density over the space, but it can be generalized to discrete or bounded data distributions.

Denoising Score Matching (DSM)

1. The Score Matching objective requires several regularity conditions for $\log p_{\mathrm{d}}(\mathbf{x})$:
   - it should be continuously differentiable
   - it should be finite everywhere
2. These conditions may not always hold in practice, such as distribution of gray level of pixels in images.
3. The distribution of digital images is typically discrete and bounded.
4. Therefore, $\log p_{\mathrm{d}}(\mathbf{x})$ is discontinuous and is negative infinity outside the range, and therefore **SM is not directly applicable**.
5. To alleviate this difficulty, one can add a bit of noise to each data point: $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$
6. As long as the noise distribution $p(\epsilon)$ is smooth, the resulting noisy data distribution $q(\tilde{\mathbf{x}}) = \int q(\tilde{\mathbf{x}} \mid \mathbf{x})\, p_{\mathrm{d}}(\mathbf{x}) d\mathbf{x}$ is also smooth.
7. Thus the Fisher divergence $\mathrm{D}_{FS}(\, q(\tilde{\mathbf{x}}) \,||\, p_\theta(\tilde{\mathbf{x}}))$ is a proper objective.

1. It has been shown that the objective with noisy data can be approximated by **the noiseless Score Matching objective plus a regularization term**.
2. This regularization makes Score Matching applicable to a wider range of data distributions, but **still requires expensive second-order derivatives**.
3. One elegant and scalable solution to the above difficulty, is to show

$$\begin{aligned} \mathrm{D}_{FS}(\, q(\tilde{\mathbf{x}}) \,||\, p_\theta(\tilde{\mathbf{x}})) &= \mathbb{E}_{q(\tilde{\mathbf{x}})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log\, q(\tilde{\mathbf{x}}) - \nabla_{\mathbf{x}} \log\, p_\theta(\tilde{\mathbf{x}})\|_2^2\right] \\ &= \mathbb{E}_{q(\tilde{\mathbf{x}},\mathbf{x})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log\, q(\tilde{\mathbf{x}} \mid \mathbf{x}) - \nabla_{\mathbf{x}} \log\, p_\theta(\tilde{\mathbf{x}})\|_2^2\right] + \textit{Constant} \end{aligned}$$

4. The above expectation is again approximated by the empirical average of samples, thus completely avoiding both the unknown term $p_{\mathrm{d}}(\mathbf{x})$ and computationally expensive second-order derivatives.
5. This estimation method is called **Denoising Score Matching** (DSM) (Vincent 2011).

1. The major drawback of adding noise to data arises when $p_{\text{d}}(\mathbf{x})$ is already a well-behaved distribution that satisfies the regularity conditions required by Score Matching.
2. In this case, $\mathrm{D}_{FS}(\, q(\tilde{\mathbf{x}}) \mid\mid \, p_\theta(\tilde{\mathbf{x}})) \neq \mathrm{D}_{FS}(\, p_{\text{d}}(\mathbf{x}) \mid\mid \, p_\theta(\mathbf{x}))$, and **DSM is not a consistent objective** because the optimal EBM matches the noisy distribution $q(\tilde{\mathbf{x}})$ not $p_{\text{d}}(\mathbf{x})$.
3. This inconsistency becomes **non-negligible** when $q(\tilde{\mathbf{x}})$ significantly differs from $p_{\text{d}}(\mathbf{x})$.
4. One way to attenuate the inconsistency of DSM is to choose $q(\mathbf{x}) \approx p_{\text{d}}(\mathbf{x})$.
5. This often significantly increases the variance of objective values and hinders optimization.
6. For example, suppose $q(\tilde{\mathbf{x}} \mid \mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 \mathbf{I})$, where $\sigma \approx 0$. The corresponding DSM objective is

$$\begin{aligned}\mathrm{D}_{FS}(\, q(\tilde{\mathbf{x}}) \mid\mid \, p_\theta(\tilde{\mathbf{x}})) &= \mathbb{E}_{p_{\text{d}}(\mathbf{x})}\left[\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}\left[\frac{1}{2}\left\|\frac{\mathbf{z}}{\sigma} + \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}+\sigma\mathbf{z})\right\|_2^2\right]\right] \\ &\approx \frac{1}{2m}\sum_{i=1}^{m}\left\|\frac{\mathbf{z}^{(i)}}{\sigma} + \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}^{(i)}+\sigma\mathbf{z}^{(i)})\right\|_2^2\end{aligned}$$

where $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ are some iid samples from $p_{\text{d}}(\mathbf{x})$.

1. When $\sigma \to 0$, we can leverage Taylor series expansion to rewrite the Monte Carlo estimator to obtain

$$\mathrm{D}_{FS}(\, q(\tilde{\mathbf{x}}) \,||\, p_\theta(\tilde{\mathbf{x}})) \approx \frac{1}{2m} \sum_{i=1}^{m} \left\{ \frac{2}{\sigma} \left(\mathbf{z}^{(i)}\right)^{\intercal} \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}^{(i)}) + \frac{\left\|\mathbf{z}^{(i)}\right\|_2^2}{\sigma^2} \right\} + \textit{Constant}$$

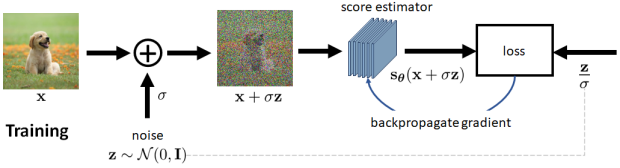
2. When estimating the above expectation with samples, the variances of $\left(\mathbf{z}^{(i)}\right)^{\intercal} \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}^{(i)})/\sigma$ and $\frac{\left\|\mathbf{z}^{(i)}\right\|_2^2}{\sigma^2}$ will both grow unbounded as $\sigma \to 0$ due to division by σ and σ^2.
3. This enlarges the variance of DSM and makes optimization challenging.
4. Some methods were proposed to solve this issue.

Let $\mathbf{x}$ be a training data and we corrupt the training data using Gaussian noise, then the corrupted version will be $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$.

Thus, we have the following relation for $\nabla_{\mathbf{x}} \log\, q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$.

$$
\begin{aligned}
\nabla_{\tilde{\mathbf{x}}} \log\, q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) &= \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I}) \\
&= \nabla_{\tilde{\mathbf{x}}} \log \frac{\exp\!\left(-\frac{1}{2}(\tilde{\mathbf{x}} - \mathbf{x})^\intercal \cdot \left(\sigma^2 \mathbf{I}\right)^{-1} \cdot (\tilde{\mathbf{x}} - \mathbf{x})\right)}{\sqrt{(2\pi)^d (\sigma^2 \mathbf{I})}} \\
&= \nabla_{\tilde{\mathbf{x}}} \left[-\frac{1}{2\sigma^2} (\tilde{\mathbf{x}} - \mathbf{x})^\intercal \cdot \mathbf{I} \cdot (\tilde{\mathbf{x}} - \mathbf{x}) \right] - \underbrace{\nabla_{\tilde{\mathbf{x}}} \log \sqrt{(2\pi)^d (\sigma^2 \mathbf{I})}}_{=0} \\
&= -\frac{1}{2\sigma^2} \nabla_{\tilde{\mathbf{x}}} [(\tilde{\mathbf{x}} - \mathbf{x})^\intercal \cdot (\tilde{\mathbf{x}} - \mathbf{x})] \\
&= -\frac{1}{\sigma^2} (\tilde{\mathbf{x}} - \mathbf{x}) = \frac{1}{\sigma^2} (\mathbf{x} - \tilde{\mathbf{x}}) \approx \mathbf{s}(\tilde{\mathbf{x}}).
\end{aligned}
$$

Since the network outputs $\mathbf{s}_\theta(\tilde{\mathbf{x}})$, then it is evident why this method is called *denoising*.



Denoising Score Matching (DSM)

1. The denoising score matching uses a fixed noise level σ, which leads to much of the input space unexplored.
2. Instead in Langevin dynamics, the score network, $\mathbf{s}_\theta(\mathbf{x})$, is trained to handle a variety of discrete noise levels (Song and Ermon 2019).
3. Let noise levels $\{\sigma_1, \sigma_2, \ldots, \sigma_T\}$ be a decreasing geometric sequences such that $\frac{\sigma_1}{\sigma_2} = \frac{\sigma_2}{\sigma_3} = \ldots, \frac{\sigma_{T-1}}{\sigma_T} > 1$.
4. The loss function for training the score network is

$$\begin{aligned}\mathcal{J}_{dsm}(\theta, \sigma_t) &= \underbrace{\sigma_t^2}_{\text{Loss weight}} \mathbb{E}_{\mathbf{x}, \tilde{\mathbf{x}} \sim q_{\sigma_t}(\mathbf{x}, \tilde{\mathbf{x}})} \left[\| \mathbf{s}_\theta(\mathbf{x}) - \nabla_{\tilde{\mathbf{x}}} \log\, q_{\sigma_t}(\mathbf{x}, \tilde{\mathbf{x}}) \|^2 \right] \\ &= \mathbb{E}_{\mathbf{x}, \tilde{\mathbf{x}} \sim q_{\sigma_t}(\mathbf{x}, \tilde{\mathbf{x}})} \left[\left\| \sigma_t \mathbf{s}_\theta(\tilde{\mathbf{x}}) - \left(\frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma_t} \right) \right\|^2 \right]\end{aligned}$$

5. At training time, the scale of loss is roughly equal across σ_t, because $\frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma_t} \sim \mathcal{N}(0, \mathbf{I})$.
6. Also, it is empirically found that $\|\mathbf{s}_\theta(\mathbf{x})\|_2 \propto \frac{1}{\sigma}$.
7. At inference time, they used η_t instead of η, where η_t is given by

$$\eta_t = \eta \left(\frac{\sigma_t}{\sigma_T} \right)^2.$$

Sliced Score Matching (SSM)

1. By adding noise to data, DSM avoids the expensive computation of second-order derivatives.
2. However, DSM does not give a consistent estimator of the data distribution.
3. In order to use score matching for learning deep energy-based models, we have to compute $\|\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})\|_2^2$ and $\operatorname{tr}\big(\nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\big)$.
   - Term $\|\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})\|_2^2$ can be computed by one simple backpropagation of $E_\theta(\mathbf{x})$.
   - Term $\operatorname{tr}\big(\nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\big)$ requires much more number of backpropagations to compute.
   - Computing $\operatorname{tr}\big(\nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\big)$ requires a number of backpropagation that is proportional to the data dimension D (Martens, Sutskever, and Swersky 2012).
4. Therefore, score matching is not scalable when learning deep energy-based models on high-dimensional data.
5. **Sliced Score Matching** is an alternative to Denoising Score Matching that is **both consistent and computationally efficient** (Song, Garg, et al. 2019).

1. The idea is that one dimensional data distribution is much easier to estimate for score matching.
2. Song et. al. proposed to project the scores onto random directions, such that the vector fields of scores of the data and model distribution become scalar fields (Song, Garg, et al. 2019).
3. Then comparing the scalar fields to determine how far the model distribution is from the data distribution.
4. Two vector fields are equivalent if and only if their scalar fields corresponding to projections onto all directions are the same.
5. Let $\mathbf{v}$ be a random projection direction and $p_{\mathbf{v}}(\mathbf{x})$ as its distribution.
6. The random projected version of Fisher divergence is

$$\mathrm{D}_{SF}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x})) = \frac{1}{2}\, \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\Big[(\mathbf{v}^\intercal \nabla_{\mathbf{x}} \log\, p_{\mathrm{d}}(\mathbf{x}) - \mathbf{v}^\intercal \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}))^2\Big]$$

called sliced Fisher divergence.

1. Unfortunately, sliced Fisher divergence has the same problem as Fisher divergence, due to the unknown data score function $\nabla_{\mathbf{x}} \log\, p_{\mathrm{d}}(\mathbf{x})$.
2. By using integration by parts, we obtain the following tractable alternative form

$$\mathrm{D}_{SF}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x})) = \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\left[\mathbf{v}^\intercal \nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\mathbf{v} + \frac{1}{2}(\mathbf{v}^\intercal \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}))^2\right] + \text{Constant}$$

   - Term $\mathbf{v}^\intercal \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})$ can be computed by one backpropagation for deep energy-based models.
   - Term $\mathbf{v}^\intercal \nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x})\mathbf{v}$ involves Hessian, but it is in the form of Hessian-vector products, which can be computed within $O(1)$ backpropagations.
3. Therefore, the computation of sliced score matching does not depend on the dimension of data, and is much more scalable for training deep energy-based models on high dimensional datasets.

1. Instead of minimizing the Fisher divergence between two vector-valued scores, SSM randomly samples a projection vector $\mathbf{v}$, takes the inner product between $\mathbf{v}$ and the two scores, and then compare the resulting two scalars.
2. Sliced Score Matching minimizes the following divergence called the sliced Fisher divergence

$$\mathrm{D}_{SF}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x})) = \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\left[\mathbb{E}_{p_{\mathbf{v}}(\mathbf{v})}\left[\frac{1}{2}\sum_{i=1}^{D}\left(\frac{\partial E_\theta(\mathbf{x})}{\partial x_i} v_i\right)^2 + \sum_{i=1}^{D}\sum_{j=1}^{D}\frac{\partial^2 E_\theta(\mathbf{x})}{\partial x_i \partial x_j} v_i v_j\right]\right] + \textit{Constant}$$

3. All expectations in the above objective can be estimated with empirical means.

1. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be iid samples from the data distribution $p_{\mathrm{d}}(\mathbf{x})$.
2. For each $\mathbf{x}_i$, draw M random projection directions $\{\mathbf{v}_{i1}, \ldots, \mathbf{v}_{iM}\} \sim p_{\mathbf{v}}(\mathbf{v})$.
3. The sliced score matching objective can be estimated with empirical averages, giving rise to the following finite-sample estimator:

$$\frac{1}{mM} \sum_{i=1}^{m} \sum_{j=1}^{M} \left\{ \mathbf{v}_{ij}^{\mathsf{T}} \nabla_{\mathbf{x}}^2 \log\, p_\theta(\mathbf{x}_i) \mathbf{v}_{ij} + \frac{1}{2} \left(\mathbf{v}_{ij}^{\mathsf{T}} \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}_i) \right)^2 \right\}$$

4. Let $\widehat{\theta}_{mM}$ be the minimizer of the above empirical estimator, and let θ^* be the true parameter corresponding to the data distribution such that $p_{\theta^*}(\mathbf{x}) = p_{\mathrm{d}}(\mathbf{x})$.
5. It has been shown that under some regularity conditions, $\widehat{\theta}_{mM}$ is consistent and asymptotically normal.
6. Formally, for any $M \in \mathbb{N}^+$, when $m \to \infty$, we have

$$\widehat{\theta}_{mM} \xrightarrow{p} \theta^*$$
$$\sqrt{m}\left(\widehat{\theta}_{mM} - \theta^*\right) \xrightarrow{d} \mathcal{N}(0, \Sigma)$$

where Σ is some covariance matrix.

References

Reading

1. Paper Learning Deep Generative Models (Salakhutdinov 2015).
2. Chapter A Tutorial on Energy-Based Learning (Lecun et al. 2006).
3. Paper How to Train Your Energy-Based Models (Song and Kingma 2021).
4. Chapter 24 of Probabilistic Machine Learning: Advanced Topics (Murphy 2023).
5. Section 9.2 of Deep Generative Modeling (Tomczak 2024).

References i

- Che, Tong et al. (2020). "Your GAN is Secretly an Energy-based Model and You Should Use Discriminator Driven Latent Sampling". In: *Advances in Neural Information Processing Systems*.
- Hyvärinen, Aapo (2005). "Estimation of Non-Normalized Statistical Models by Score Matching". In: *Journal of Machine Learning Research* 6, pp. 695–709.
- Lecun, Yann et al. (2006). "A tutorial on energy-based learning". In: *Predicting structured data*. Ed. by G. Bakir et al. MIT Press.
- Martens, James, Ilya Sutskever, and Kevin Swersky (2012). "Estimating the Hessian by Back-propagating Curvature". In: *International Conference on Machine Learning*.
- Murphy, Kevin P. (2023). *Probabilistic Machine Learning: Advanced Topics*. The MIT Press.
- Salakhutdinov, Ruslan (2015). "Learning Deep Generative Models". In: *Annual Review of Statistics and Its Application* 2, pp. 361–385.
- Song, Yang and Stefano Ermon (2019). "Generative Modeling by Estimating Gradients of the Data Distribution". In: *Advances in Neural Information Processing Systems*, pp. 11895–11907.
- Song, Yang, Sahaj Garg, et al. (2019). "Sliced Score Matching: A Scalable Approach to Density and Score Estimation". In: *Uncertainty in Artificial Intelligence*. Vol. 115, pp. 574–584.
- Song, Yang and Diederik P. Kingma (2021). "How to Train Your Energy-Based Models". In: *CoRR* abs/2101.03288.

References ii

- Tomczak, Jakub M. (2024). *Deep Generative Modeling*. Springer.
- Vincent, Pascal (2011). "A Connection Between Score Matching and Denoising Autoencoders". In: *Neural Comput.* 23.7, pp. 1661–1674.
- Xiao, Zhisheng, Qing Yan, and Yali Amit (2020). "Exponential Tilting of Generative Models: Improving Sample Quality by Training and Sampling from Latent Energy". In: *CoRR* abs/2006.08100.

Questions?