

---

# A Note on Score Matching Langevin Dynamics

---

J. S.

2025.05.03

## Abstract

This is a personal summary of the mathematical foundations of *score matching Langevin dynamics*.

## Contents

<b>1</b>	<b>Mathematical Foundations of Score Matching Langevin Dynamics</b>	<b>1</b>
1.1	Score Matching . . . . .	1
1.2	Denoising Score Matching . . . . .	3
1.3	Langevin Dynamics . . . . .	4
1.3.1	Langevin Equation in Physics . . . . .	4
1.3.2	Langevin Dynamics in Generative Models . . . . .	4
<b>2</b>	<b>Appendix</b>	<b>6</b>
2.1	Proof of Implicit Score Matching . . . . .	6
2.2	Parzen Density Estimate . . . . .	7
2.3	Denoising Autoencoder . . . . .	7
2.4	Fokker-Planck Equation . . . . .	8
2.4.1	Definition . . . . .	8
2.4.2	Fokker-Planck Equation for Langevin Dynamics . . . . .	9

## 1 Mathematical Foundations of Score Matching Langevin Dynamics

### 1.1 Score Matching

Assume we have a continuous random variable  $\mathbf{x} \in \mathbb{R}^D$  with some unknown distribution  $p(\mathbf{x})$ . We want to use a parameterized model  $p_\theta(\mathbf{x})$  to approximate the true distribution  $p(\mathbf{x})$ . What we have is a set of i.i.d. samples  $\{\mathbf{x}_i\}_{i=1}^N$  from the distribution.

One way to model the distribution is to estimate the **score**, i.e., the gradient of the log-density function with respect to the data. Intuitively, the score is a vector that points in the direction of the steepest increase of the log density.

**Note.** This use differs slightly from traditional statistics terminology where score usually refers to the derivative of the log likelihood with respect to parameters, whereas here we are talking about a score with respect to the data. The score we use is called the **Stein score**.

We denote the real score function as

$$\psi(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) \quad (1.1)$$

and the estimated score function as

$$\mathbf{s}(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\mathbf{x}} \log p_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{bmatrix} s_1(\mathbf{x}) \\ \vdots \\ s_D(\mathbf{x}) \end{bmatrix}. \quad (1.2)$$

We use  $\mathbf{s}(\mathbf{x})$  instead of  $\mathbf{s}(\mathbf{x}; \boldsymbol{\theta})$  to denote the estimated score function in the following for simplicity.

We can define the **squared distance** between the estimated and the true score functions as:

$$J(\boldsymbol{\theta}) = \frac{1}{2} \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \|\mathbf{s}(\boldsymbol{\xi}) - \boldsymbol{\psi}(\boldsymbol{\xi})\|^2 d\boldsymbol{\xi}. \quad (1.3)$$

Thus, our **score matching** estimator of  $\boldsymbol{\theta}$  is given by

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}).$$

However, we do not know the true score function  $\boldsymbol{\psi}(\mathbf{x})$ . In practice, we can use non-parametric probability kernel density estimation methods (Parzen window estimation) to get an approximated score function  $\hat{\boldsymbol{\psi}}(\mathbf{x})$ , and minimize the squared distance between  $\mathbf{s}(\mathbf{x})$  and  $\hat{\boldsymbol{\psi}}(\mathbf{x})$  instead. This approach is called **Explicit Score Matching (ESM)**.

Instead of explicitly matching the true score, **Implicit Score Matching** [Hyvärinen, 2005] is proposed, which avoids the need for a target. Expanding the formula of squared distance, we have

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{2} \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \left[ \|\mathbf{s}(\boldsymbol{\xi})\|^2 - 2\mathbf{s}(\boldsymbol{\xi})^T \boldsymbol{\psi}(\boldsymbol{\xi}) + \|\boldsymbol{\psi}(\boldsymbol{\xi})\|^2 \right] d\boldsymbol{\xi} \\ &= \underbrace{\left[ \frac{1}{2} \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \|\mathbf{s}(\boldsymbol{\xi})\|^2 d\boldsymbol{\xi} \right]}_{\text{Term 1}} + \underbrace{\left[ - \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \mathbf{s}(\boldsymbol{\xi})^T \boldsymbol{\psi}(\boldsymbol{\xi}) d\boldsymbol{\xi} \right]}_{\text{Term 2}} + \underbrace{\left[ \frac{1}{2} \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \|\boldsymbol{\psi}(\boldsymbol{\xi})\|^2 d\boldsymbol{\xi} \right]}_{\text{constant}}. \end{aligned} \quad (1.4)$$

Here

$$\text{Term 1} = \frac{1}{2} \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \left[ \sum_{i=1}^D s_i(\boldsymbol{\xi})^2 \right] d\boldsymbol{\xi},$$

$$\begin{aligned}
\text{Term 2} &= - \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \sum_{i=1}^D s_i(\boldsymbol{\xi}) \psi_i(\boldsymbol{\xi}) d\boldsymbol{\xi} \\
&= - \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \sum_{i=1}^D s_i(\boldsymbol{\xi}) \frac{\partial \log p(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi} \\
&= - \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \sum_{i=1}^D s_i(\boldsymbol{\xi}) \frac{1}{p(\boldsymbol{\xi})} \frac{\partial p(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi} \\
&= - \int_{\boldsymbol{\xi} \in \mathbb{R}^D} \sum_{i=1}^D s_i(\boldsymbol{\xi}) \frac{\partial p(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi} \\
&= - \sum_{i=1}^D \int_{\boldsymbol{\xi} \in \mathbb{R}^D} s_i(\boldsymbol{\xi}) \frac{\partial p(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi} \\
&= \sum_{i=1}^D \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \frac{\partial s_i(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi}.
\end{aligned}$$

**Note.** See Appendix for the proof of the last line.

So we have

$$J(\boldsymbol{\theta}) = \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \sum_{i=1}^D \left[ \frac{1}{2} s_i(\boldsymbol{\xi})^2 + \frac{\partial s_i(\boldsymbol{\xi})}{\partial \xi_i} \right] d\boldsymbol{\xi} + \text{constant}. \quad (1.5)$$

We have thus proven the remarkable fact that our objective no longer requires having an explicit score target.

In practice, we have  $N$  observations of the random vector  $\mathbf{x}$ , denoted by  $\mathbf{x}_1, \dots, \mathbf{x}_N$ . The sample version of  $J$  is obtained as

$$\tilde{J}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{t=1}^N \sum_{i=1}^D \left[ \frac{1}{2} s_i(\mathbf{x}_t)^2 + \frac{\partial s_i(\mathbf{x}_t)}{\partial (\mathbf{x}_t)_i} \right] + \text{constant}, \quad (1.6)$$

which is asymptotically equivalent to  $J$  due to the law of large numbers. By minimizing  $\tilde{J}$ , we obtain  $\hat{\boldsymbol{\theta}}$  and the corresponding estimator of the score function,  $\mathbf{s}(\mathbf{x}; \hat{\boldsymbol{\theta}})$ .

**Note.** The score matching objective can also be written as

$$\begin{aligned}
J(\boldsymbol{\theta}) &= \int_{\mathbf{x} \in \mathbb{R}^D} p(\mathbf{x}) \left[ \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})) + \frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|_2^2 \right] d\mathbf{x} + \text{constant} \\
&= \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \left[ \text{tr}(\nabla_{\mathbf{x}} \mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})) + \frac{1}{2} \|\mathbf{s}_{\boldsymbol{\theta}}(\mathbf{x})\|_2^2 \right] + \text{constant}.
\end{aligned}$$

## 1.2 Denoising Score Matching

**Denoising Score Matching (DSM)** [Vincent, 2011] is inspired by both the Score Matching principle and the Denoising Autoencoder (DAE) [Vincent et al., 2008] approach.

We denote the clean sample as  $\mathbf{x}$ , and get the corrupted sample  $\tilde{\mathbf{x}}$  by a pre-defined perturbation  $q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})$ . Denoising score matching defines the following objective:

$$\mathcal{J}_{\text{DSM}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}, \tilde{\mathbf{x}} \sim q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})} \left[ \|\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_{\sigma}(\tilde{\mathbf{x}}|\mathbf{x})\|^2 \right]. \quad (1.7)$$

The underlying intuition is that following the gradient of the log density at some corrupted point,  $\tilde{\mathbf{x}}$  should ideally move toward the clean sample  $\mathbf{x}$ .

Note that if we choose the gaussian kernel  $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 \mathbf{I})$ , we have

$$\nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}}).$$

The direction  $\frac{1}{\sigma^2}(\mathbf{x} - \tilde{\mathbf{x}})$  clearly corresponds to moving from  $\tilde{\mathbf{x}}$  back toward clean sample  $\mathbf{x}$ , and we want  $\mathbf{s}_\theta(\tilde{\mathbf{x}})$  to match that as best it can. Thus, with a **Gaussian kernel**, the DSM objective has a simpler form:

$$\mathcal{J}_{\text{DSM}}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})} \left[ \left\| \mathbf{s}_\theta(\tilde{\mathbf{x}}) - \frac{\mathbf{x} - \tilde{\mathbf{x}}}{\sigma^2} \right\|^2 \right]. \quad (1.8)$$

Minimizing the objective, we obtain a score estimator  $\mathbf{s}_\theta(\tilde{\mathbf{x}})$  which gives the direction that should be followed to move from  $\tilde{\mathbf{x}}$  toward  $\mathbf{x}$ . With some proper sampling algorithm, we can generate samples of  $p_{\text{data}}$  starting from any initial point  $\mathbf{x}_0$ .

## 1.3 Langevin Dynamics

### 1.3.1 Langevin Equation in Physics

**Langevin equation** is a stochastic differential equation (SDE) describing how a system evolves when subjected to a combination of deterministic and fluctuating (“random”) forces.

The original Langevin equation describes **Brownian motion**. The form is given by

$$m \frac{d\mathbf{v}}{dt} = -\lambda \mathbf{v} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, 2\lambda k_B T \mathbf{I}). \quad (1.9)$$

Here,  $\mathbf{v}$  is the velocity of the particle,  $\lambda$  is its damping coefficient,  $m$  is its mass;  $k_B$  is the Boltzmann constant,  $T$  is the temperature. The force acting on the particle is written as a sum of

1. a viscous force proportional to the particle’s velocity;
2. a random force  $\boldsymbol{\eta}(t)$  representing the effect of the collisions with the molecules of the fluid.

The force  $\boldsymbol{\eta}(t)$  has a Gaussian probability distribution with correlation function

$$\langle \boldsymbol{\eta}(t), \boldsymbol{\eta}(t') \rangle = 2\lambda k_B T \mathbf{I} \cdot \delta(t - t')$$

This indicates that the random force is time-independent.

### 1.3.2 Langevin Dynamics in Generative Models

If we consider other external forces, they can all be represented by a potential force  $-\nabla_{\mathbf{x}} U(\mathbf{x})$ . The Langevin equation can then be written as

$$m \frac{d\mathbf{v}}{dt} = -\nabla U(\mathbf{x}) - \lambda \mathbf{v} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, 2\lambda k_B T \mathbf{I}). \quad (1.10)$$

Sometimes the inertia (mass) of the particle is negligible in comparison with the damping force. For instance, the pollen in a viscous fluid can be considered as a particle with zero inertia. In this case, the

trajectory of the particle is described by the so-called **Overdamped Langevin Equation** [Sandrić, 2018]:

$$0 = -\nabla U(\mathbf{x}) - \lambda \mathbf{v} + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, 2\lambda k_B T \mathbf{I}), \quad (1.11)$$

or equivalently,

$$\lambda d\mathbf{x} = -\nabla U(\mathbf{x})dt + \sqrt{2\lambda k_B T} \boldsymbol{\epsilon} dt, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

In domains other than physics, we often set  $\lambda = 1$ ,  $T = 1$ ,  $k_B = 1$  for simplicity. Then we have

$$d\mathbf{x} = -\nabla U(\mathbf{x})dt + \sqrt{2} \boldsymbol{\epsilon} dt, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

If we set the potential as the negative log probability density, i.e.,  $U(\mathbf{x}) = -\log p(\mathbf{x})$ , we get the *overdamped Langevin equation in generative modeling*:

$$d\mathbf{x} = \nabla \log p(\mathbf{x})dt + \sqrt{2} \boldsymbol{\epsilon} dt, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (1.12)$$

The discrete form is given as

$$\begin{aligned} \frac{\mathbf{x}_{t+1} - \mathbf{x}_t}{\eta} &= \nabla \log p(\mathbf{x}_t) + \sqrt{2} \boldsymbol{\epsilon}_t, \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ \mathbf{x}_{t+1} - \mathbf{x}_t &= \eta \nabla \log p(\mathbf{x}_t) + \sqrt{2\eta} \mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \end{aligned} \quad (1.13)$$

This is the iteration formula used in generative modeling.

By thermodynamics, the equilibrium distribution of  $\mathbf{x}$  is the **Boltzmann distribution**

$$p_{\text{equilibrium}}(\mathbf{x}) = \exp \left[ -\frac{U(\mathbf{x})}{k_B T} \right] / Z$$

where  $Z$  is a normalizing constant. Since we already set  $\lambda = 1$ ,  $T = 1$ ,  $k_B = 1$ , which leads to  $U(\mathbf{x})/k_B T = -\log p(\mathbf{x})$ , we have

$$p_{\text{equilibrium}}(\mathbf{x}) = p(\mathbf{x}).$$

So if we choose the negative log probability density as the potential, and iteratively update  $\mathbf{x}$  with overdamped Langevin dynamics,  $\mathbf{x}$  will eventually converge to a equilibrium distribution  $p_{\text{equilibrium}}(\mathbf{x})$ , which is exactly the pre-set probability distribution  $p(\mathbf{x})$ .

In other words, for any probability distribution  $p(\mathbf{x})$ , as long as we know its **score**, i.e.,  $\nabla p(\mathbf{x})$ , we can generate samples from it using overdamped Langevin dynamics.

## References

- Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL <http://jmlr.org/papers/v6/hyvarinen05a.html>.
- Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23:1661–1674, 2011. doi: 10.1162/NECO\_a.00142. URL [https://doi.org/10.1162/NECO\\_a\\_00142](https://doi.org/10.1162/NECO_a_00142).
- Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008. URL <https://dl.acm.org/doi/abs/10.1145/1390156.1390294>.

Nikola Sandrić. Stability of the overdamped langevin equation in double-well potential, 2018. URL <https://arxiv.org/abs/1807.10922>.

Stanley H. Chan. Tutorial on diffusion models for imaging and vision, 2025. URL <https://arxiv.org/abs/2403.18103>.

## 2 Appendix

### 2.1 Proof of Implicit Score Matching

**Proposition 2.1.** Assume  $f : \mathbb{R}^D \rightarrow \mathbb{R}$  and  $g : \mathbb{R}^D \rightarrow \mathbb{R}$  are both scalar-valued continuous differentiable functions defined on  $\mathbb{R}^D$ . Additionally, assume  $f(\boldsymbol{\xi})$  and  $g(\boldsymbol{\xi})$  both approaches 0 when  $\|\boldsymbol{\xi}\|$  approaches infinity. Then we have

$$0 = \int_{\boldsymbol{\xi} \in \mathbb{R}^D} f(\boldsymbol{\xi}) \frac{\partial g(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi} + \int_{\boldsymbol{\xi} \in \mathbb{R}^D} g(\boldsymbol{\xi}) \frac{\partial f(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi}$$

for all  $i \in \{1, 2, \dots, D\}$ .

*Proof.* By the product rule, we have

$$\frac{\partial [f(\boldsymbol{\xi})g(\boldsymbol{\xi})]}{\partial \xi_1} = f(\boldsymbol{\xi}) \frac{\partial g(\boldsymbol{\xi})}{\partial \xi_1} + g(\boldsymbol{\xi}) \frac{\partial f(\boldsymbol{\xi})}{\partial \xi_1}$$

Take the integral on the left side, we have

$$\begin{aligned} \int_{\boldsymbol{\xi} \in \mathbb{R}^D} \frac{\partial [f(\boldsymbol{\xi})g(\boldsymbol{\xi})]}{\partial \xi_1} d\boldsymbol{\xi} &= \int_{\xi_2, \xi_3, \dots, \xi_D} \left[ \int_{\xi_1 \in \mathbb{R}} \frac{\partial [f(\boldsymbol{\xi})g(\boldsymbol{\xi})]}{\partial \xi_1} d\xi_1 \right] d\xi_2 d\xi_3 \dots d\xi_D \\ &= \int_{\xi_2, \xi_3, \dots, \xi_D} \left[ f(+\infty, \xi_2, \xi_3, \dots, \xi_D) g(+\infty, \xi_2, \xi_3, \dots, \xi_D) \right. \\ &\quad \left. - f(-\infty, \xi_2, \xi_3, \dots, \xi_D) g(-\infty, \xi_2, \xi_3, \dots, \xi_D) \right] d\xi_2 d\xi_3 \dots d\xi_D \\ &= \int_{\xi_2, \xi_3, \dots, \xi_D} [0 \cdot 0 - 0 \cdot 0] d\xi_2 d\xi_3 \dots d\xi_D \\ &= 0 \end{aligned}$$

Take the integral on the right side, we have

$$0 = \int_{\boldsymbol{\xi} \in \mathbb{R}^D} f(\boldsymbol{\xi}) \frac{\partial g(\boldsymbol{\xi})}{\partial \xi_1} d\boldsymbol{\xi} + \int_{\boldsymbol{\xi} \in \mathbb{R}^D} g(\boldsymbol{\xi}) \frac{\partial f(\boldsymbol{\xi})}{\partial \xi_1} d\boldsymbol{\xi}.$$

The same applies for all indices of  $\xi_i$ . □

For implicit score matching, by the definitions in Section 1.1,  $p(\boldsymbol{\xi})$  and  $s_i(\boldsymbol{\xi})$  are continuous differentiable functions from  $\mathbb{R}^D$  to  $\mathbb{R}$ . Meanwhile, when  $\xi_i$  approaches infinity,  $p(\boldsymbol{\xi})$  and  $s_i(\boldsymbol{\xi})$  approach 0. By Proposition 2.1, we have

$$\int_{\boldsymbol{\xi} \in \mathbb{R}^D} s_i(\boldsymbol{\xi}) \frac{\partial p(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi} = - \int_{\boldsymbol{\xi} \in \mathbb{R}^D} p(\boldsymbol{\xi}) \frac{\partial s_i(\boldsymbol{\xi})}{\partial \xi_i} d\boldsymbol{\xi}.$$

Thus we have proved the transformation of the second term in (1.4).

## 2.2 Parzen Density Estimate

The **Parzen density estimate**, also known as **Parzen window method** or **kernel density estimation (KDE)**, is a non-parametric technique for estimating the probability density function (PDF) of a random variable from a finite set of data points.

Consider a random variable  $X \in \mathbb{R}^d$ . We want to estimate its probability density function  $f(\mathbf{x})$  using a set of observations  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ . Parzen window method gives the estimated density as

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{h} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right), \quad (2.1)$$

where  $K$  is the kernel function (normalized to integrate to 1) and  $h$  is the *bandwidth* (a hyperparameter).

**Note.** It is easy to check that  $\hat{f}(\mathbf{x})$  integrates to 1. Denote  $\mathbf{u} = (\mathbf{x} - \mathbf{x}_i)/h$ , then  $d\mathbf{x} = h d\mathbf{u}$ , and we have

$$\begin{aligned} \int_{\mathbf{x} \in \mathbb{R}^d} \hat{f}(\mathbf{x}) d\mathbf{x} &= \frac{1}{n} \sum_{i=1}^n \int_{\mathbf{x} \in \mathbb{R}^d} \frac{1}{h} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) d\mathbf{x} \\ &= \frac{1}{n} \sum_{i=1}^n \int_{\mathbf{u} \in \mathbb{R}^d} \frac{1}{h} K(\mathbf{u}) h d\mathbf{u} \\ &= \frac{1}{n} \sum_{i=1}^n \underbrace{\int_{\mathbf{u} \in \mathbb{R}^d} K(\mathbf{u}) d\mathbf{u}}_{=1} \\ &= \frac{1}{n} \cdot n = 1. \end{aligned}$$

Common kernels include:

- **Gaussian:**

$$K(\mathbf{u}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{\|\mathbf{u}\|_2^2}{2}\right);$$

- **Epanechnikov (parabolic):**

$$K(\mathbf{u}) = \begin{cases} \frac{d+2}{2c_d} (1 - \|\mathbf{u}\|_2^2), & \|\mathbf{u}\|_2 \leq 1 \\ 0, & \text{otherwise} \end{cases};$$

- **Uniform:**

$$K(\mathbf{u}) = \begin{cases} \frac{1}{c_d}, & \|\mathbf{u}\|_2 \leq 1 \\ 0, & \text{otherwise} \end{cases},$$

where  $c_d = \frac{\pi^{d/2}}{\Gamma(1 + \frac{d}{2})}$  is the volume of the  $d$ -dimensional unit ball.

## 2.3 Denoising Autoencoder

**Denoising autoencoder (DAE)** [Vincent et al., 2008] is a simple modification of classical autoencoder neural networks that are trained not to reconstruct their input but rather to denoise an artificially corrupted version of their input.

Let us consider some dataset  $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^N$  consisting of  $N$  i.i.d. samples of some continuous or discrete random variable  $\mathbf{x} \in \mathbb{R}^d$ . Following the logic of autoencoder, the denoising process can be formulated as:

1. Input: denoised sample  $\tilde{\mathbf{x}}$ ;
2. Pass  $\tilde{\mathbf{x}}$  through the encoder to obtain the latent variable  $\mathbf{z}$ ;
3. Pass  $\mathbf{z}$  through the decoder to obtain the reconstructed data  $\mathbf{x}'$ .

Usually there is no need to separate the encoder and decoder, as they can be combined into a single network. We denote the network as  $f_\theta : \mathbb{R}^d \rightarrow \mathbb{R}^d$ ,  $\tilde{\mathbf{x}} \mapsto \mathbf{x}'$ .

The training process becomes:

1. Sample an observation  $\mathbf{x}$ ;
2. Corrupt the observation with some noise  $\epsilon$  to create  $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$  (usually  $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ );
3. Denoise the noisy observation  $\tilde{\mathbf{x}}$  to obtain  $\mathbf{x}' = f_\theta(\tilde{\mathbf{x}})$ ;
4. Update the parameters  $\theta$  using some loss with  $\mathbf{x}$  and  $\mathbf{x}'$ .

## 2.4 Fokker-Planck Equation

### 2.4.1 Definition

**Fokker-Planck equation** is a partial differential equation that describes the time evolution of the probability density function of a particle under the influence of drag forces and random forces, as in Brownian motion. It is also known as the **Kolmogorov forward equation**.

- **One Dimensional Case:** Consider an Itô process:

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \quad (2.2)$$

where  $\mu(X_t, t)$  is the drift,  $D(X_t, t) = \sigma^2(X_t, t)/2$  is the diffusion coefficient, and  $W_t$  is a standard Wiener process. The Fokker-Planck equation for the probability density  $p(x, t)$  of the random variable  $X_t$  is

$$\frac{\partial}{\partial t} p(x, t) = -\frac{\partial}{\partial x} [\mu(x, t)p(x, t)] + \frac{\partial^2}{\partial x^2} [D(x, t)p(x, t)]. \quad (2.3)$$

- **Multi-Dimensional Case:** Consider an Itô process in  $N$  dimensions:

$$d\mathbf{X}_t = \boldsymbol{\mu}(\mathbf{X}_t, t)dt + \boldsymbol{\sigma}(\mathbf{X}_t, t)d\mathbf{W}_t, \quad (2.4)$$

where  $\mathbf{X}_t$  and  $\boldsymbol{\mu}(\mathbf{X}_t, t)$  are  $N$ -dimensional vectors,  $\boldsymbol{\sigma}(\mathbf{X}_t, t)$  is an  $N \times M$  matrix, and  $\mathbf{W}_t$  is an  $M$ -dimensional standard Wiener process. The probability density  $p(\mathbf{x}, t)$  for  $\mathbf{X}_t$  satisfies the multi-dimensional Fokker-Planck equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\sum_{i=1}^N \frac{\partial}{\partial x_i} [\mu_i(\mathbf{x}, t)p(\mathbf{x}, t)] + \sum_{i=1}^N \sum_{j=1}^N \frac{\partial^2}{\partial x_i \partial x_j} [D_{ij}(\mathbf{x}, t)p(\mathbf{x}, t)], \quad (2.5)$$

where  $\boldsymbol{\mu} = (\mu_1, \dots, \mu_N)$  is the drift vector, and  $\mathbf{D} = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\sigma}^T$  is the diffusion tensor.



### 2.4.2 Fokker-Planck Equation for Langevin Dynamics

In generative modeling, assume the pre-set probability distribution is  $p_r(\mathbf{x})$ , the SDE of Langevin Dynamics is given by

$$d\mathbf{X}_t = \nabla \log p_r(\mathbf{x})dt + \sqrt{2}d\mathbf{W}_t. \quad (2.6)$$

Matching Fokker-Planck equation's form, we have

$$\boldsymbol{\mu}(\mathbf{x}, t) = \nabla \log p_r(\mathbf{x}),$$

$$\boldsymbol{\sigma}(\mathbf{x}, t) = \sqrt{2}\mathbf{I},$$

$$\mathbf{D}(\mathbf{x}, t) = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\sigma}^T = \mathbf{I},$$

$$\mu_i(\mathbf{x}, t) = \frac{\partial}{\partial x_i} \log p_r(\mathbf{x}),$$

$$D_{ij}(\mathbf{x}, t) = \delta_{ij}.$$

The evolution of the probability density function is given by

$$\begin{aligned} \frac{\partial p(\mathbf{x}, t)}{\partial t} &= - \sum_{i=1}^N \frac{\partial}{\partial x_i} \left[ \left( \frac{\partial}{\partial x_i} \log p_r(\mathbf{x}) \right) p(\mathbf{x}, t) \right] + \sum_{i=1}^N \sum_{j=1}^N \frac{\partial^2}{\partial x_i \partial x_j} \left[ \delta_{ij} p(\mathbf{x}, t) \right] \\ &= - \sum_{i=1}^N \frac{\partial}{\partial x_i} \left[ \left( \frac{1}{p_r(\mathbf{x})} \frac{\partial}{\partial x_i} p_r(\mathbf{x}) \right) p(\mathbf{x}, t) \right] + \sum_{i=1}^N \frac{\partial^2}{\partial x_i^2} \left[ p(\mathbf{x}, t) \right] \\ &= - \sum_{i=1}^N \frac{\partial}{\partial x_i} \left[ \frac{p(\mathbf{x}, t)}{p_r(\mathbf{x})} \frac{\partial}{\partial x_i} p_r(\mathbf{x}) \right] + \sum_{i=1}^N \frac{\partial^2}{\partial x_i^2} \left[ p(\mathbf{x}, t) \right] \\ &= - \nabla \cdot \left[ \frac{p(\mathbf{x}, t)}{p_r(\mathbf{x})} \nabla p_r(\mathbf{x}) \right] + \nabla^2 p(\mathbf{x}, t). \end{aligned}$$

Assume we have already reached the stationary state  $p(\mathbf{x}, t) = p_r(\mathbf{x})$ , then  $\frac{\partial p(\mathbf{x}, t)}{\partial t} = 0$ , which means the probability distribution  $p(\mathbf{x}, t)$  converges to the target distribution  $p_r(\mathbf{x})$  when  $t \rightarrow \infty$ .