Diffusion Model and Score Matching

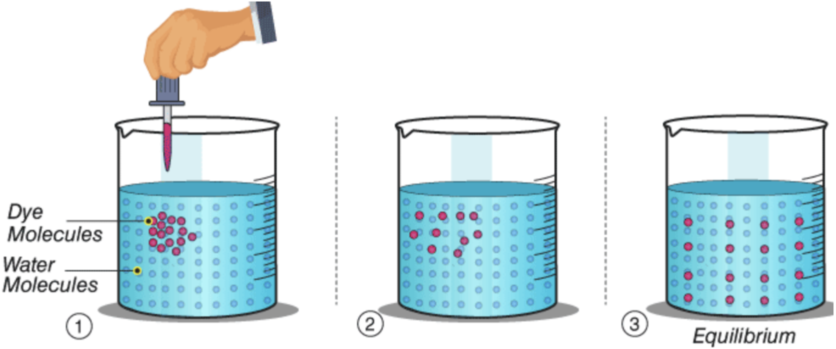
Instructor: Lei Wu [1]

Mathematical Introduction to Machine Learning

Peking University, Fall 2024

[1]School of Mathematical Sciences; Center for Machine Learning Research

What Is Diffusion?



Dye molecules **diffuse** throughout the entire space by colliding with water molecules.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].

[2] η depends on the temperature, time unit, etc.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].
- Thus, we have after N steps

$$x_{N\eta} = x_0 + \sqrt{\eta}\sum_{k=0}^{N-1}\xi_k \sim \mathcal{N}(x_0, \eta N).$$

[2] η depends on the temperature, time unit, etc.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].

- Thus, we have after N steps

$$x_{N\eta} = x_0 + \sqrt{\eta}\sum_{k=0}^{N-1} \xi_k \sim \mathcal{N}(x_0, \eta N).$$

- Consider the continuous-time limit: $\eta \to 0$. Let $t = N\eta$. Then, we have

$$x_{N\eta} \to X_t \sim \mathcal{N}(X_0, t).$$

[2] η depends on the temperature, time unit, etc.

Mathematical Model of Diffusion: Brownian motion

- Let $\{x_k\}_{k\geq 0}$ be the trajectory of dye molecules. We can model its dynamics as follows

$$x_{k+1} = x_k + \sqrt{\eta}\xi_k, \qquad 0 \leq k \leq N-1,$$

where $\xi_k \overset{iid}{\sim} \mathcal{N}(0,1)$ and η is a small factor [2].

- Thus, we have after N steps

$$x_{N\eta} = x_0 + \sqrt{\eta}\sum_{k=0}^{N-1}\xi_k \sim \mathcal{N}(x_0, \eta N).$$

- Consider the continuous-time limit: $\eta \to 0$. Let $t = N\eta$. Then, we have

$$x_{N\eta} \to X_t \sim \mathcal{N}(X_0, t).$$

- We call $B_t := X_t - X_0$ Brownian motion.

[2] η depends on the temperature, time unit, etc.

Important Properties of Brownian Motion

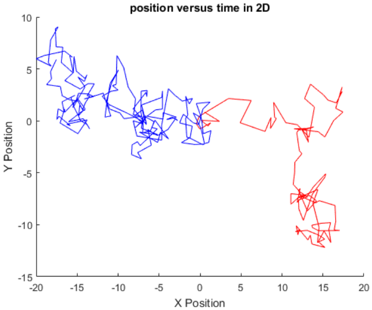


Figure 1: A animation of Brownian motion: https://physics.bu.edu/~duffy/HTML5/brownian_motion.html

[3] Albert Einstein, On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, 1905.

Important Properties of Brownian Motion

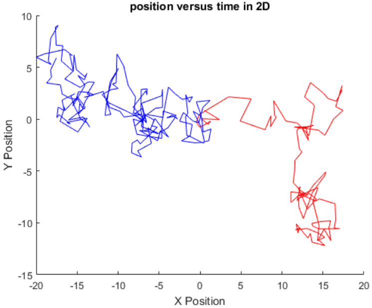


Figure 1: A animation of Brownian motion: https://physics.bu.edu/~duffy/HTML5/brownian_motion.html

- After time t, dye molecules only move $O(\sqrt{t})$: [3]

$$\mathbb{E}[B_t] = 0, \quad \mathbb{E}[B_t^2] = t,$$

where the second property is known as the Einstein relationship.

[3] Albert Einstein, On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, 1905.

Important Properties of Brownian Motion

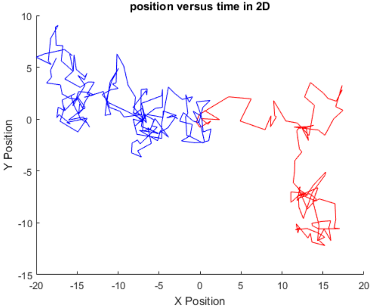


Figure 1: A animation of Brownian motion: https://physics.bu.edu/~duffy/HTML5/brownian_motion.html

- After time t, dye molecules only move $O(\sqrt{t})$: [3]

$$\mathbb{E}[B_t] = 0, \quad \mathbb{E}[B_t^2] = t,$$

  where the second property is known as the Einstein relationship.
- $B_t - B_s$ and B_s are independent. The trajectory is continuous but **non-differentiable almost everywhere**.

[3] Albert Einstein, On the motion of small particles suspended in liquids at rest required by the molecular-kinetic theory of heat, 1905.

General Diffusion Process (Modeled by Ito-SDE)

Consider dye molecules in a force field $f(x,t)$ and the collision is heterogeneous:

- From t to $t+\eta$, the dye molecule moves according to

$$x_{t+\eta} - x_t = \underbrace{f(x_t,t)\eta}_{\text{drift}} + \underbrace{\sigma(x_t,t)\sqrt{\eta}\xi_t}_{\text{diffusion}}.$$

[4] A good textbook for SDE is: Bernt Øksendal, Stochastic Differential Equations: An Introduction with Applications

General Diffusion Process (Modeled by Ito-SDE)

Consider dye molecules in a force field $f(x, t)$ and the collision is heterogeneous:

- From t to $t + \eta$, the dye molecule moves according to

$$x_{t+\eta} - x_t = \underbrace{f(x_t, t)\eta}_{\text{drift}} + \underbrace{\sigma(x_t, t)\sqrt{\eta}\xi_t}_{\text{diffusion}}.$$

- Taking $\eta \to 0$ gives a stochastic differential equation (SDE) [4]:

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + \sigma(x_t, t)\,\mathrm{d}B_t$$

[4] A good textbook for SDE is: Bernt Øksendal, Stochastic Differential Equations: An Introduction with Applications

General Diffusion Process (Modeled by Ito-SDE)

Consider dye molecules in a force field $f(x, t)$ and the collision is heterogeneous:

- From t to $t + \eta$, the dye molecule moves according to

$$x_{t+\eta} - x_t = \underbrace{f(x_t, t)\eta}_{\text{drift}} + \underbrace{\sigma(x_t, t)\sqrt{\eta}\xi_t}_{\text{diffusion}}.$$

- Taking $\eta \to 0$ gives a stochastic differential equation (SDE) [4]:

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + \sigma(x_t, t)\,\mathrm{d}B_t$$

  In physics, it is often written (by let $\omega_t = \dot{B}_t$) as

$$\dot{x}_t = f(x_t, t) + \sigma(x_t, t)\omega_t,$$

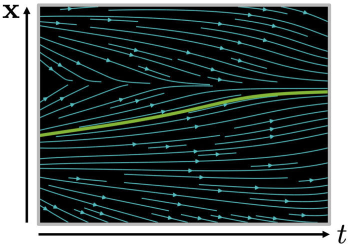
  where ω_t is often referred to as white noise.

[4] A good textbook for SDE is: Bernt Øksendal, Stochastic Differential Equations: An Introduction with Applications

A Comparison Between SDE and ODE [5]

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

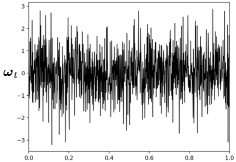
Iterative Numerical Solution:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

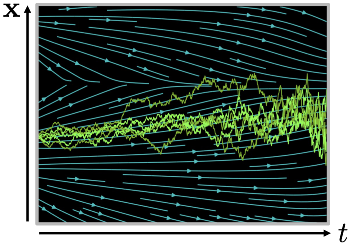
Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)}_{\text{diffusion coefficient}}\boldsymbol{\omega}_t$$

Wiener Process (Gaussian White Noise)



$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$



$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\mathcal{N}(\mathbf{0}, \mathbf{I})$$

[5] taken from https://cvpr2022-tutorial-diffusion-models.github.io/

Langevin Dynamics

- (Over-damped) **Langevin dynamics** is a special SDE with the drift term given by a potential force $f(x) = -\nabla U(x)$:

$$\mathrm{d}x_t = -\nabla U(x_t)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B_t. \tag{1}$$

  Denote by $p_t = p(\cdot, t) = \mathrm{Law}(X_t)$. Then, we have

$$p(x,t) \to \frac{e^{-\beta U(x)}}{Z_\beta} \text{ as } t \to \infty. \tag{2}$$

Langevin Dynamics

- (Over-damped) **Langevin dynamics** is a special SDE with the drift term given by a potential force $f(x) = -\nabla U(x)$:

$$\mathrm{d}x_t = -\nabla U(x_t)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B_t. \tag{1}$$

  Denote by $p_t = p(\cdot, t) = \mathrm{Law}(X_t)$. Then, we have

$$p(x,t) \to \frac{e^{-\beta U(x)}}{Z_\beta} \text{ as } t \to \infty. \tag{2}$$

- To simulate (1), we can apply the Euler-Maruyama scheme:

$$X_{k+1} = X_k - \nabla U(X_k)\eta + \sqrt{2\beta^{-1}\eta}\,\xi_k \ \text{ with } \ \xi_k \sim \mathcal{N}(0, I_d). \tag{3}$$

Langevin Dynamics

- (Over-damped) **Langevin dynamics** is a special SDE with the drift term given by a potential force $f(x) = -\nabla U(x)$:

$$\mathrm{d}x_t = -\nabla U(x_t)\,\mathrm{d}t + \sqrt{2\beta^{-1}}\,\mathrm{d}B_t. \tag{1}$$

Denote by $p_t = p(\cdot, t) = \mathrm{Law}(X_t)$. Then, we have

$$p(x,t) \to \frac{e^{-\beta U(x)}}{Z_\beta} \text{ as } t \to \infty. \tag{2}$$

- To simulate (1), we can apply the Euler-Maruyama scheme:

$$X_{k+1} = X_k - \nabla U(X_k)\eta + \sqrt{2\beta^{-1}\eta}\xi_k \text{ with } \xi_k \sim \mathcal{N}(0, I_d). \tag{3}$$

- **Ornstein–Uhlenbeck** (OU) process is a simplest SDE given by

$$\mathrm{d}x_t = -\theta x_t\,\mathrm{d}t + \sigma\,\mathrm{d}B_t,$$

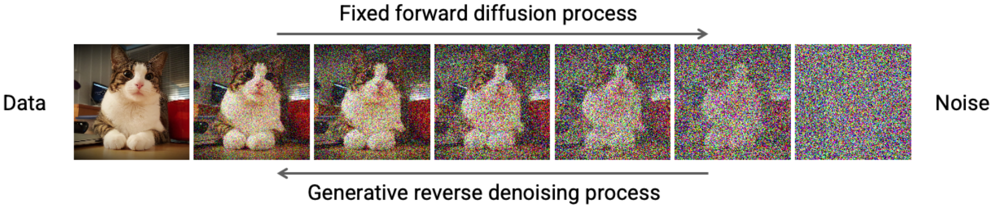
for which $U(x) = \theta\|x\|^2/2, \beta^{-1} = \sigma^2/2$. The equilibrium distribution is Gaussian:

$$p_\infty(x) \propto \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$$

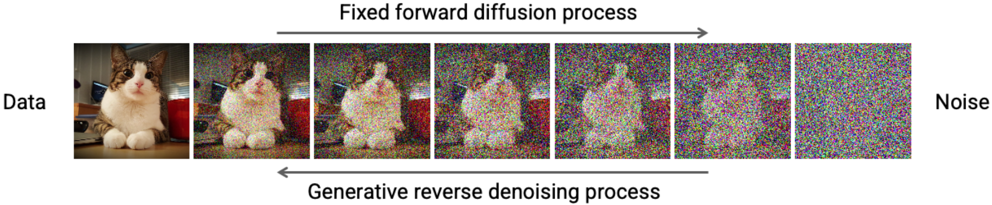
Diffusion models

In diffusion models

- We first gradually inject noise to a sample until it becomes pure noise. **This is a diffusion process**!!
- The generative models are (probabilistic) inverse of the forward process.



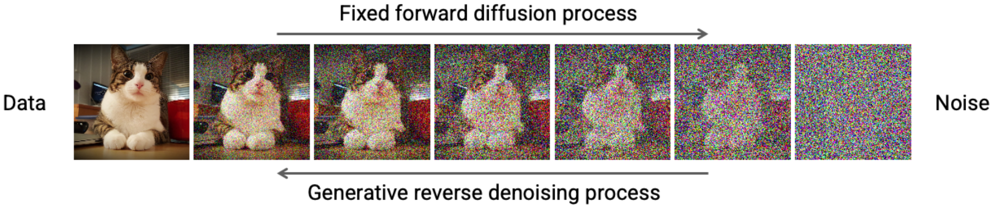
Diffusion models



Why are diffusion models powerful?

- Guide the learning of reverse **generative** denoise process with the information of a **fixed forward diffusion process**!
- GAN, Normalizing flow, and Variational Autoencoder do not have forward-process information to guide the learning. [**Explain it!**]

Diffusion models



There are two key issues in diffusion models:

- Construct forward diffusion process.
- Utilize forward information for learning the reverse process.

Denoising Diffusion Probabilistic Models (DDPM) [6]

- DDPM chooses the following **variance-preserving** forward diffusion process:

$$x_{k+1} = \sqrt{1-\beta_k}x_k + \sqrt{\beta_k}\xi_k, 0 \le k \le N-1,$$

where $\xi_k \overset{iid}{\sim} \mathcal{N}(0, I_d)$.

[6] Jonathan Ho, Ajay Jain, Pieter Abbeel, *Denoising Diffusion Probabilistic Models*, NeurIPS 2020.

Denoising Diffusion Probabilistic Models (DDPM) [6]

- DDPM chooses the following **variance-preserving** forward diffusion process:

$$x_{k+1} = \sqrt{1-\beta_k} x_k + \sqrt{\beta_k}\xi_k, 0 \le k \le N-1,$$

  where $\xi_k \overset{iid}{\sim} \mathcal{N}(0, I_d)$.
- Consider $\beta_t = \beta = o(1)$. Then, we have

$$x_{k+1} = x_k - \frac{\beta x_k}{2} + \sqrt{\beta}\xi_k + o(\beta) \tag{4}$$

  When $\beta \to 0$, we have the forward process is given by an OU process

$$\mathrm{d}x_t = -\frac{x_t}{2}\,\mathrm{d}t + \mathrm{d}B_t.$$

[6] Jonathan Ho, Ajay Jain, Pieter Abbeel, *Denoising Diffusion Probabilistic Models*, NeurIPS 2020.

Properties of the Forward Process

- First, the conditional distribution is always Gaussian

$$P_t := x_t | x_0 \sim \mathcal{N}(e^{-t/2} x_0, (1 - e^{-t}) I_d) = \mathcal{N}\left(\alpha_t x_0, \sqrt{1 - \alpha_t^2} I_d\right), \tag{5}$$

where $\alpha_t = e^{-t/2}$. We also denote $\sigma_t^2 := 1 - \alpha_t^2$. (Derivation is given on the blackboard.)

Properties of the Forward Process

- First, the conditional distribution is always Gaussian

$$P_t := x_t | x_0 \sim \mathcal{N}(e^{-t/2} x_0, (1 - e^{-t}) I_d) = \mathcal{N}\left(\alpha_t x_0, \sqrt{1 - \alpha_t^2} I_d\right), \tag{5}$$

  where $\alpha_t = e^{-t/2}$. We also denote $\sigma_t^2 := 1 - \alpha_t^2$. (Derivation is given on the blackboard.)
- The distribution of x_t can be viewed as the convolution of $P(x_0)$ with a Gaussian smoothing kernel:

$$P_t(x) = \int P_t(x | x_0) P(x_0) \, \mathrm{d}x_0 = \int P(x_0) \frac{1}{C_t} e^{-\frac{\|x - \alpha_t x_0\|^2}{2(1 - \alpha_t^2)}} \, \mathrm{d}x_0,$$

  where C_t is the normalizing constant.

Properties of the Forward Process

- First, the conditional distribution is always Gaussian

$$P_t := x_t | x_0 \sim \mathcal{N}(e^{-t/2} x_0, (1 - e^{-t}) I_d) = \mathcal{N}\left(\alpha_t x_0, \sqrt{1 - \alpha_t^2} I_d\right), \tag{5}$$

  where $\alpha_t = e^{-t/2}$. We also denote $\sigma_t^2 := 1 - \alpha_t^2$. (Derivation is given on the blackboard.)
- The distribution of x_t can be viewed as the convolution of $P(x_0)$ with a Gaussian smoothing kernel:

$$P_t(x) = \int P_t(x|x_0) P(x_0) \,\mathrm{d}x_0 = \int P(x_0) \frac{1}{C_t} e^{-\frac{\|x - \alpha_t x_0\|^2}{2(1-\alpha_t^2)}} \,\mathrm{d}x_0,$$

  where C_t is the normalizing constant.
- The forward process converges exponentially fast:

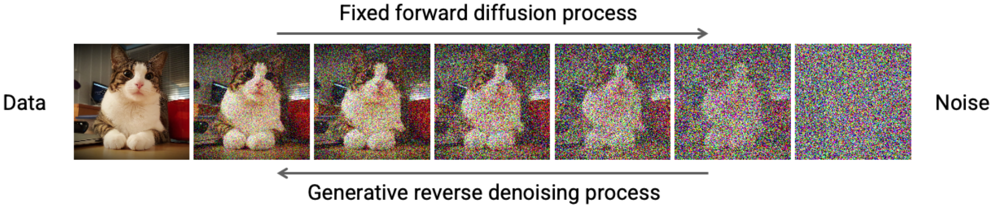
$$D_{\mathrm{KL}}\left(P_t || \mathcal{N}(0, I_d)\right) \leq C e^{-t} D_{\mathrm{KL}}\left(P_0 || \mathcal{N}(0, I_d)\right),$$

  This means we can take a moderately large T such that

$$\mathrm{Law}(x_T) \approx \mathcal{N}(0, I_d).$$

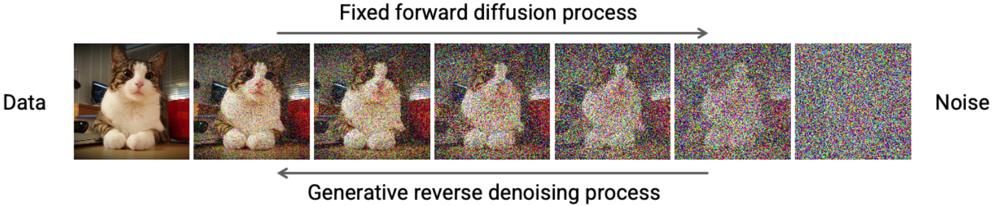
Reversing a Diffusion Process

What do we mean by reversing a diffusion process?



Reversing a Diffusion Process

What do we mean by reversing a diffusion process?



Definition 1

Given a forward process $\{X_t\}_{t\in[0,T]}$, the backward process $\{\tilde{X}_t\}_{t\in[T,0]}$ is said to a reverse process of $\{X_t\}_{t\in[0,T]}$ iff

$$\mathrm{Law}(X_t) = \mathrm{Law}(\tilde{X}_{T-t}).$$

Remark: The reverse process may be non-unique.

An Explicit Construction of Reverse Processes

- Consider a large family of diffusion process given by the forward SDE:

$$\mathrm{d}x_t = f(x,t)\,\mathrm{d}t + g(t)\,\mathrm{d}B_t, \quad 0 \leq t \leq T.$$

[7] Brian Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982)

An Explicit Construction of Reverse Processes

- Consider a large family of diffusion process given by the forward SDE:

$$\mathrm{d}x_t = f(x,t)\,\mathrm{d}t + g(t)\,\mathrm{d}B_t, \quad 0 \le t \le T.$$

- **Anderson (1982)** [7] provided an explicit construction of the reverse SDE:

$$\mathrm{d}\tilde{x}_t = \left[f(\tilde{x}_t,t) - g^2(t)\nabla_x \log p(\tilde{x},t)\right]\mathrm{d}t + g(t)\,\mathrm{d}\bar{B}_t, \quad t \in [T,0]$$

where $\bar{B}_t$ is a backward Brownian motion and the time in the above equation is negative. (The proof can be easily completed by checking the Fokker-Planck equation (omitted). We refer to Anderson (1982) for the derivation.)

[7] Brian Anderson, *Reverse-time diffusion equation models*, Stochastic Processes and their Applications 12 (1982)

Score Matching

- The key quantity for the reverse SDE is the (time-dependent) **score function**

$$\nabla_x \log p(\cdot, \cdot) : \mathbb{R}^d \times [0, T] \mapsto \mathbb{R}^d,$$

Therefore, **we can easily reverse the process if the score function is known**.

Score Matching

- The key quantity for the reverse SDE is the (time-dependent) **score function**

$$\nabla_x \log p(\cdot, \cdot) : \mathbb{R}^d \times [0, T] \mapsto \mathbb{R}^d,$$

  Therefore, **we can easily reverse the process if the score function is known**.
- **Model:** Let $s_\theta : \mathbb{R}^d \times [0, T] \mapsto \mathbb{R}^d$ be a neural network to model the score function.

Score Matching

- The key quantity for the reverse SDE is the (time-dependent) **score function**

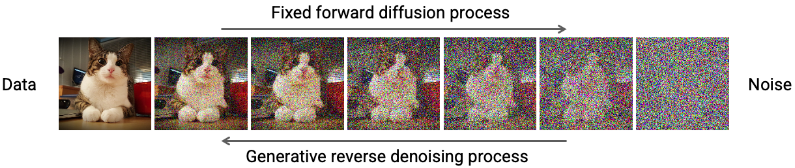
$$\nabla_x \log p(\cdot,\cdot) : \mathbb{R}^d \times [0,T] \mapsto \mathbb{R}^d,$$

  Therefore, **we can easily reverse the process if the score function is known**.
- **Model:** Let $s_\theta : \mathbb{R}^d \times [0,T] \mapsto \mathbb{R}^d$ be a neural network to model the score function.
- **Training objective:** Let $p_t = p(\cdot, t)$ and

$$L_t(\theta) = \mathbb{E}_{x\sim p_t}\left[\|s_\theta(x,t) - \nabla_x \log p(x,t)\|^2\right].$$

  Let π be a (weighted) distribution supported on $[0,T]$. Consider the learning via

$$\min_\theta L(\theta) := \mathbb{E}_{t\sim\pi}[L_t(\theta)] \qquad \textbf{(score matching)}. \tag{6}$$



Score Matching (Cont'd)

- Why is this objective informative for training? The problem nearly becomes a sequential of supervised learning: Score matching at different times.
- **Bad News:** $\nabla_x \log p(\cdot, t)$ is unknown. Instead, we have only access to the noisy sequences $\{x_i(t)\}_{t\in[0,T], i\in[n]}$ generated by the forward process, starting from the inputs $\{x_i(0) = x_i\}_{i=1}^n$.
- **Approach:** Reformulate the objective into a quantity that computes an expectation with respect to $p(\cdot, t)$ (This a general principle!).

Implicit Score Matching

Reformulate the objective using the **log-derivative trick**:

$$
\begin{aligned}
L_t(\theta) &= \mathbb{E}_{x\sim p_t}\left[\|s_\theta(x,t) - \nabla_x \log p(x,t)\|^2\right] \\
&= \mathbb{E}_{x\sim p_t}\|s_\theta(x,t)\|^2 + \mathbb{E}_{x\sim p_t}\|\nabla_x \log p(x,t)\|^2 - 2\mathbb{E}_{x\sim p_t}\langle s_\theta(x,t), \nabla_x \log p(x,t)\rangle \\
&= \mathbb{E}_{x\sim p_t}\|s_\theta(x,t)\|^2 + \mathbb{E}_{x\sim p_t}\|\nabla_x \log p(x,t)\|^2 - 2\int_{\mathbb{R}^d}\langle s_\theta(x,t), \nabla_x p(x,t)\rangle\,\mathrm{d}x \\
&= \mathbb{E}_{x\sim p_t}\|s_\theta(x,t)\|^2 + \mathbb{E}_{x\sim p_t}\|\nabla_x \log p(x,t)\|^2 + 2\int_{\mathbb{R}^d}[\nabla_x \cdot s_\theta(x,t)]p(x,t)\,\mathrm{d}x.
\end{aligned}
$$

Note that the conditional distribution $p_t(x|z)$ is tractable (see Eq. (5)). However, there are a few problems with the above formula:

- Computing gradient for the red term is **computationally expensive**, as $\nabla_\theta(\nabla_x \cdot s_\theta(x,t))$ needs to compute second-order derivatives.
- Stochastic approximation also exhibits a high variance, as during the training $\mathrm{Var}_x[\nabla_x \cdot s_\theta(x,t)]$ is not well-controlled.

We need a better alternative.

Denoising Score Matching

Noting that $p(x,t) = \int p_t(x|z) p_0(z)\,\mathrm{d}z$, we have

$$\begin{aligned}
\int_{\mathbb{R}^d} [\nabla \cdot s_\theta(x,t)] p(x,t)\,\mathrm{d}x &= \int_{\mathbb{R}} p_0(z)\,\mathrm{d}z \int_{\mathbb{R}^d} [\nabla_x \cdot s_\theta(x,t)] p_t(x|z)\,\mathrm{d}x \\
&= -\int_{\mathbb{R}} p_0(z)\,\mathrm{d}z \int_{\mathbb{R}^d} \langle s_\theta(x,t), \nabla_x p_t(x|z)\rangle\,\mathrm{d}x \\
&= -\int_{\mathbb{R}} p_0(z)\,\mathrm{d}z \int_{\mathbb{R}^d} \langle s_\theta(x,t), \nabla_x \log p_t(x|z)\rangle p_t(x|z)\,\mathrm{d}x \\
&= -\,\mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\left[\langle s_\theta(x,t), \nabla_x \log p_t(x|z)\rangle\right]
\end{aligned}$$

Based on the preceding derivation, we have

$$\begin{aligned}
L_t(\theta) &= \mathbb{E}_{x\sim p_t}\left[\|s_\theta(x,t) - \nabla_x \log p(x,t)\|^2\right] \\
&= \mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\,\|s_\theta(x,t)\|^2 - 2\,\mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\left[\langle s_\theta(x,t), \nabla_x \log p_t(x|z)\rangle\right] + C \\
&= \mathbb{E}_{z\sim p_0}\,\mathbb{E}_{x\sim p_t(\cdot|z)}\left[\|s_\theta(x,t) - \nabla_x \log p_t(x|z)\|^2\right] + C
\end{aligned}$$

The key observation:

- In this formula, the input gradient term $\nabla_x \log p_t(x|z)$ is explicit, eliminating the need for backpropagation and making it well-controlled.

The Denoising/Noise-Prediction Interpretation

Consider the DDPM-type [8] forward process and let $\alpha_t = e^{-t/2}$ and $\sigma_t^2 = 1 - e^{-t}$. Then,

$$p_t(x|x_0) \propto \exp\left(-\frac{\|x - \alpha_t x_0\|^2}{2\sigma_t^2}\right).$$

Thus, the total objective becomes

$$L(\theta) = \mathbb{E}_t\, \mathbb{E}_{x_0}\, \mathbb{E}_{x_t|x_0}\left[\left\| s_\theta(x_t, t) - \frac{x_t - \alpha_t x_0}{\sigma_t^2} \right\|^2\right] \tag{7}$$

[8] DDPM-type forward process gives an explicit conditional density.

The Denoising/Noise-Prediction Interpretation

Consider the DDPM-type [8] forward process and let $\alpha_t = e^{-t/2}$ and $\sigma_t^2 = 1 - e^{-t}$. Then,

$$p_t(x|x_0) \propto \exp\left(-\frac{\|x - \alpha_t x_0\|^2}{2\sigma_t^2}\right).$$

Thus, the total objective becomes

$$L(\theta) = \mathbb{E}_t\, \mathbb{E}_{x_0}\, \mathbb{E}_{x_t|x_0}\left[\left\|s_\theta(x_t, t) - \frac{x_t - \alpha_t x_0}{\sigma_t^2}\right\|^2\right] \tag{7}$$

Noting that $x_t|x_0 \sim \mathcal{N}(\alpha_t x_0, (1-\alpha_t)I_d)$, we can rewrite

$$\underbrace{x_t}_{\text{noisy sample}} = \alpha_t x_0 + \sqrt{1-\alpha_t^2}\xi_t = \underbrace{\alpha_t x_0}_{\text{clean sample}} + \underbrace{\sigma_t \xi_t}_{\text{noise}} \quad \text{with } \xi_t \sim \mathcal{N}(0, I_d).$$

- One can interpret $x_t - \alpha_t x_0$ as the "direction of denoising".
- Plugging it back into (7) gives the **noise-prediction** objective:

$$L(\theta) = \mathbb{E}_t\, \mathbb{E}_{x_0}\, \mathbb{E}_{\xi_t \sim \mathcal{N}(0, I_d)}\left[\left\|s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t}\right\|^2\right]$$

[8] DDPM-type forward process gives an explicit conditional density.

Training Procedure

$$L(\theta) = \mathbb{E}_t \, \mathbb{E}_{x_0} \, \mathbb{E}_{\xi_t \sim \mathcal{N}(0, I_d)} \left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2 \right]$$

Parameterize s_θ with neural networks. Then, SGD of batch size 1 updates as follows:

Algorithm

- Step 1: $t \sim \pi, x_0 \sim p_0, \xi_t \sim \mathcal{N}(0, I_d)$
- Step 2: $\alpha_t = e^{-t/2}, x_t = \alpha_t x_0 + \sqrt{1 - \alpha_t} \xi_t$
- Step 3: $\hat{L}(\theta) = \left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2$
- Step 4: $\theta_{k+1} = \theta_k - \eta \nabla_\theta \hat{L}(\theta_k)$

The Choice of Time Weighting

How to choose π?

$$L(\theta) = \mathbb{E}_{t \sim \pi} \mathbb{E}_{x_0} \mathbb{E}_{\xi_t \sim \mathcal{N}(0, I_d)} \left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2 \right]$$

Key observation: When $t \to 0$, $\sigma_t = \sqrt{1 - e^{-t}} \to 0$. Loss heavily amplified when sampling t close to 0. High variance!

[9] Take a look this note

The Choice of Time Weighting

How to choose π?

$$L(\theta) = \mathbb{E}_{t\sim\pi}\, \mathbb{E}_{x_0}\, \mathbb{E}_{\xi_t\sim\mathcal{N}(0,I_d)} \left[\left\| s_\theta(x_t, t) - \frac{\xi_t}{\sigma_t} \right\|^2 \right]$$

Key observation: When $t \to 0$, $\sigma_t = \sqrt{1-e^{-t}} \to 0$. Loss heavily amplified when sampling t close to 0. High variance!

- Training with **time cut-off** η:

$$\pi = \mathrm{Unif}([\eta, T]).$$

- Variance reduction via **importance sampling** [9]:

$$\pi(t) \propto \frac{1}{\sigma_t^2}.$$

Intuitively, we need more samples for when t is close to 0, otherwise the Monte-Carlo estimate may give a rather wrong estimate.

[9] Take a look this note

Probability-Flow ODE

The reverse SDE is given by

$$\mathrm{d}\tilde{x}_t = \left[f(\tilde{x}_t, t) - g^2(t)\nabla_x \log p(\tilde{x}, t)\right] \mathrm{d}t + g(t)\,\mathrm{d}\bar{B}_t,$$

(Song et al. 2021) showed that the following probability-flow ODE is also a reverse process

$$\mathrm{d}\tilde{x}_t = f(\tilde{x}_t, t) - \frac{1}{2}g^2(t)\nabla_x \log p(\tilde{x}, t)\,\mathrm{d}t.$$

For the DDPM-type forward process, it becomes

$$\mathrm{d}\tilde{x}_t = -\frac{1}{2}(x_t + \nabla_x \log p(\tilde{x}, t))\,\mathrm{d}t$$

The probability-flow ODE can be interpreted as a continuous-time normalizing flow (CNF).

A Schematic Comparison

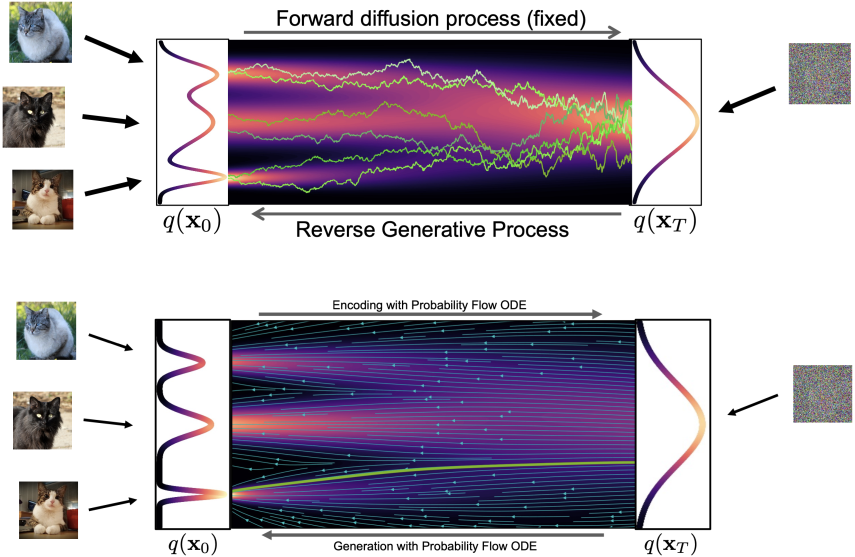


Figure 2: (Up) SDE; (Down) ODE.

Faster Sampling

- For diffusion models, generating new samples needs to discretize the revise-time SDE/ODE. It is often very slow as we need to take small step size to control discretization error and numerical stability.
  - For SDEs, in general, there does not exist higher-order solver as the trajectory is non-differentiable almost everywhere.
  - Deterministic ODE enables the use of advanced higher-order ODE solvers such as Runge-Kutta, thereby speeding up the generation of new samples.

Exact Likelihood Computation [10]

- Consider the continuous-time normalizing flow generated by the ODE

$$\dot{x}_t = f(x_t, t), t \in [0, T]$$

with initial condition $x_0 \sim p_0$.

[10] For generality, we assume normal time direction in this slide.

Exact Likelihood Computation [10]

- Consider the continuous-time normalizing flow generated by the ODE

$$\dot{x}_t = f(x_t, t), t \in [0, T]$$

  with initial condition $x_0 \sim p_0$.
- Consider the flow map $\Phi_T : \mathbb{R}^d \mapsto \mathbb{R}^d$ defined by $\Phi_T(x_0) = x_T$. Then, we have the log-likelihood of p_T satisfies (with the derivation left as homework)

$$\log p_T(x_T) = \log p_0(x_0) - \int_0^T \nabla \cdot f(x_t, t) \, \mathrm{d}t.$$

[10] For generality, we assume normal time direction in this slide.

Exact Likelihood Computation [10]

- Consider the continuous-time normalizing flow generated by the ODE

$$\dot{x}_t = f(x_t, t), t \in [0, T]$$

with initial condition $x_0 \sim p_0$.

- Consider the flow map $\Phi_T : \mathbb{R}^d \mapsto \mathbb{R}^d$ defined by $\Phi_T(x_0) = x_T$. Then, we have the log-likelihood of p_T satisfies (with the derivation left as homework)

$$\log p_T(x_T) = \log p_0(x_0) - \int_0^T \nabla \cdot f(x_t, t)\,\mathrm{d}t.$$

- In practice, for a $h : \mathbb{R}^d \mapsto \mathbb{R}^d$, $\nabla \cdot h(x)$ can be estimated using the Hutchinson-Skilling trace estimator:

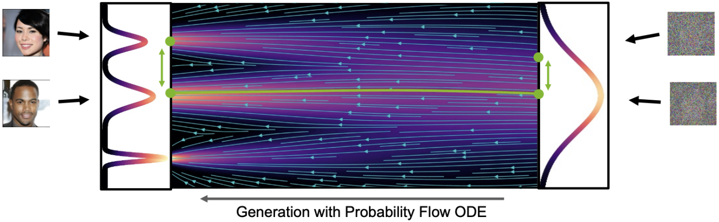
$$\underbrace{\nabla \cdot h(x)}_{\text{need } d \text{ gradients}} = \mathbb{E}_\epsilon[\epsilon^\top \nabla h(x) \epsilon] \approx \underbrace{\frac{1}{m} \sum_{j=1}^m \epsilon_j \cdot \nabla(h(x) \cdot \epsilon_j)}_{\text{need only } m \text{ gradients}},$$

where $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon\epsilon^\top] = I_d$ and $\{\epsilon_j\}_{j=1}^m$ are iid samples. The most popular choice of the distribution of ϵ is $\mathrm{Unif}(\{\pm 1\}^d)$.

[10] For generality, we assume normal time direction in this slide.

Manipulating the Latent Space

Interpolation



Controllable Generalization

Generate One-Class of Samples

Text to Images

You

请画一幅暴雪中的长城景色

ChatGPT

You

请用八大山人的风格重绘这幅画，不要改变内容，只改变风格。请注意留白和落款，不用输出任何文字描述。

ChatGPT

Some Classical Tasks

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{8}$$

where the normalizing constant disappears.

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{8}$$

where the normalizing constant disappears.

- We can use Langevin dynamics to simulate it. But we can directly couple (8) with the reverse-time SDE or ODE:

$$\begin{aligned} \dot{\tilde{x}}_t &= f(\tilde{x}_t, t) - \frac{1}{2} g(t)^2 \log p_t(\tilde{x}|y) \\ &= f(\tilde{x}_t, t) - \frac{1}{2} \nabla_x \left[\log p(\tilde{x}_t, t) + \log p(y|\tilde{x}_t) \right]. \end{aligned}$$

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{8}$$

  where the normalizing constant disappears.
- We can use Langevin dynamics to simulate it. But we can directly couple (8) with the reverse-time SDE or ODE:

$$\begin{aligned} \dot{\tilde{x}}_t &= f(\tilde{x}_t, t) - \frac{1}{2} g(t)^2 \log p_t(\tilde{x}|y) \\ &= f(\tilde{x}_t, t) - \frac{1}{2} \nabla_x \left[\log p(\tilde{x}_t, t) + \log p(y|\tilde{x}_t)\right]. \end{aligned}$$

- Therefore, as long as we have a good "classifier" $p(y|x)$, then we can couple it with the unconditional model $s_\theta(x,t) \approx \nabla \log p(x,t)$ in a very **simple and principled** approach.

Controllable Generalization via Diffusion Models

- Controlled generalization can be modeled as sampling from $P(x|y)$ where y denotes the control factor.
- Baye's rule: $P(x|y) = \frac{P(y|x)P(x)}{P(y)}$. Accordingly, the score function:

$$\nabla_x \log P(x|y) = \nabla_x \log P(y|x) + \nabla_x \log P(x) \tag{8}$$

  where the normalizing constant disappears.
- We can use Langevin dynamics to simulate it. But we can directly couple (8) with the reverse-time SDE or ODE:

$$\begin{aligned} \dot{\tilde{x}}_t &= f(\tilde{x}_t, t) - \frac{1}{2} g(t)^2 \log p_t(\tilde{x}|y) \\ &= f(\tilde{x}_t, t) - \frac{1}{2} \nabla_x \left[\log p(\tilde{x}_t, t) + \log p(y|\tilde{x}_t)\right]. \end{aligned}$$

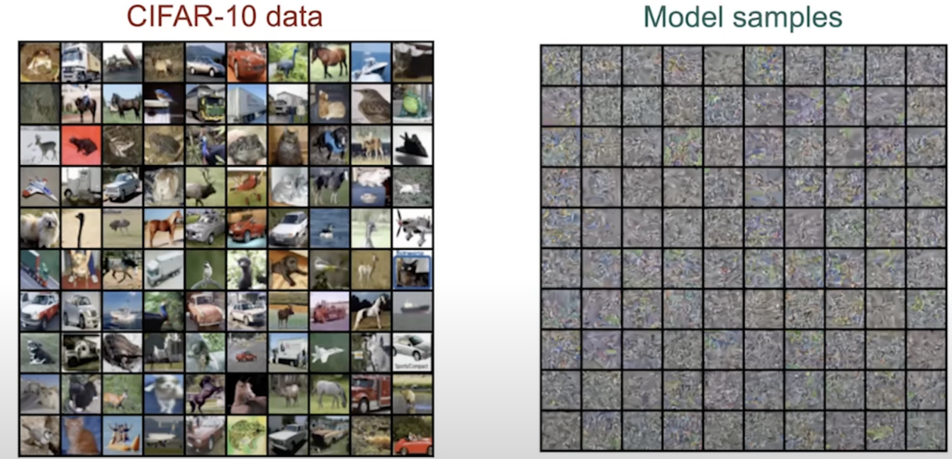
- Therefore, as long as we have a good "classifier" $p(y|x)$, then we can couple it with the unconditional model $s_\theta(x,t) \approx \nabla \log p(x,t)$ in a very **simple and principled** approach.
- **One unconditional models for all tasks.**

Connection with Energy-based Models

- In EBM, $p(x) = e^{-U(x)}/Z$. We learn a potential **energy** $V_\theta(x) \approx U(x) = -\nabla \log p(x)$. That is, score matching is also commonly used in learning energy-based model.
- In score-based models, we learn $s_\theta(x) \approx \nabla_x \log p(x) = -\nabla_x U(x)$, i.e., the **force**.
- With the score functions (aka. the force field), we can also recover samples by running Langevin dynamics
$$\mathrm{d}x_t = -\nabla_x \log p(x_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t.$$
But its performance is notorious and consequently, using the reverse-time SDE/ODE is always much better.

Naive Score Matching + Langevin Dynamics

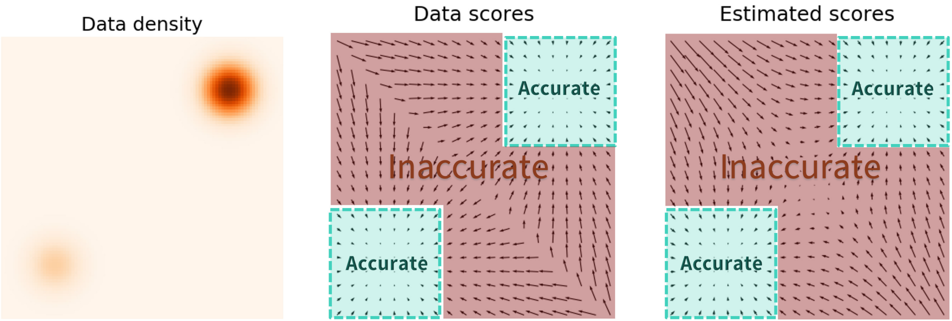
Often, the learned score function is useless when simulating Langevin dynamics.



Reason 1: Inaccurate score functions

The learned score function are inaccurate in the low-density region.

$$L(\theta) = \mathbb{E}_x \left\| s_\theta(x) - \nabla \log p(x) \right\|^2.$$



Reason 2: Sampling with Langevin Dynamics is Slow

- When the target distribution is multimodal, Langevin dynamics (or more generally, MCMC methods) struggle to efficiently sample across different modes.

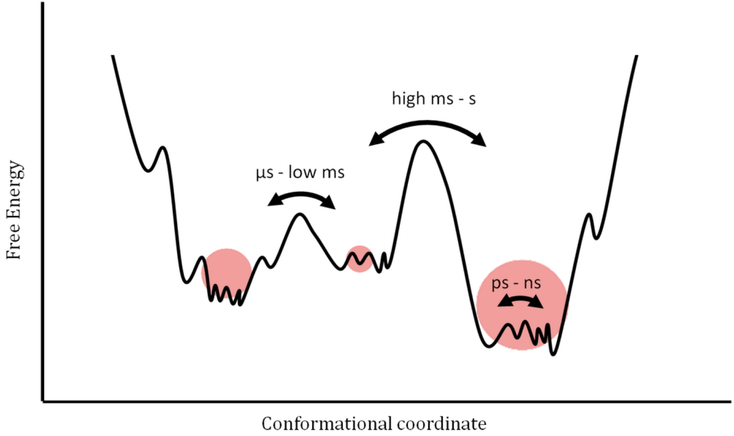


Figure 3: The sampling rate suffers from the curse of dimensionality and loss barrier.

Summary

Advantages:

- The design is principled, ensuring stable training.
- The model is highly flexible:
  - Compare with normalizing flow like real-NVP , general networks can be used to model distribution.
  - principle controllable generalization.
- Consequently, diffusion models can generate high-quality data.

Disadvantage:

- Generation is slow due to the requirement for numerous denoising steps during the "discretization" inverse process.

Summary

Advantages:

- The design is principled, ensuring stable training.
- The model is highly flexible:
  - Compare with normalizing flow like real-NVP , general networks can be used to model distribution.
  - principle controllable generalization.
- Consequently, diffusion models can generate high-quality data.

Disadvantage:

- Generation is slow due to the requirement for numerous denoising steps during the "discretization" inverse process.

Also read the good note:

- Stanley H. Chan, Tutorial on Diffusion Models for Imaging and Vision, arXv:2403.18103.