Score Matching and Flow Matching

Siqi Yao

SCHOOL OF DATA SCIENCE

June 23, 2025

Table of Contents

Table of Contents

General Scheme of Score Based Generative Modeling

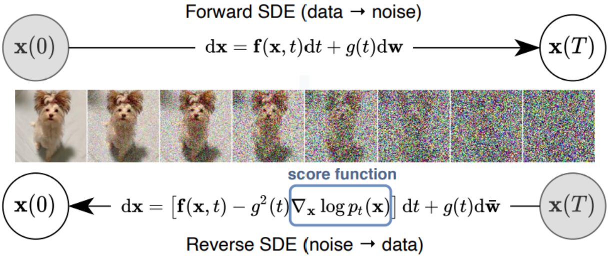

Figure 1: General Scheme. Source: [Song et al., 2020]

- **Forward process:** Gradually add noise via a forward SDE.
- **Reverse process:** Generate data by solving the reverse SDE.

General Scheme of Score Based Generative Modeling

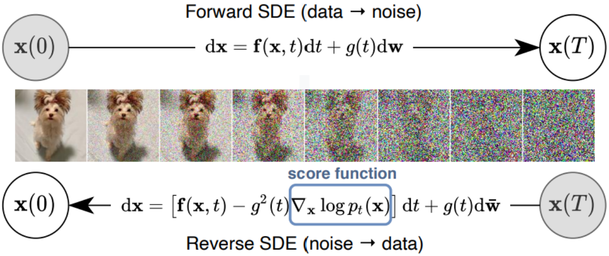

Figure 1: General Scheme. Source: [Song et al., 2020]

- **Forward process:** Gradually add noise via a forward SDE.
- **Reverse process:** Generate data by solving the reverse SDE.
- **Score estimation:** train model $\mathbf{s}_\theta(\mathbf{x}, t)$ to approximate $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$, equivalent to learning the distribution $p_t(\mathbf{x})$.

Table of Contents

Score Matching Objective

- The score function of a distribution $p_{\text{data}}(\mathbf{x})$ is defined as

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

- A **score-based model** $\mathbf{s}_\theta(\mathbf{x})$ for the score function is learned such that $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$.

Score Matching Objective

- The score function of a distribution $p_{\text{data}}(\mathbf{x})$ is defined as

$$\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$$

- A **score-based model** $\mathbf{s}_\theta(\mathbf{x})$ for the score function is learned such that $\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$.
- An straightforward training objective is the **Fisher divergence** between the model and the ground-truth score:

$$\frac{1}{2}\,\mathbb{E}_{p_{\text{data}}}\left[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2\right]$$

- However, this is intractable since $p_{\text{data}}(\mathbf{x})$ and $\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})$ are unknown.

Fisher Divergence: Equivalent Form

Theorem 1 (Equivalent transformation of Fisher divergence)

Under some weak regularity conditions, the Fisher divergence objective

$$\frac{1}{2}\,\mathbb{E}_{p_{\text{data}}}\left[\|\nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2\right]$$

is equivalent (up to a constant) to the following expression:

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

where $\nabla_{\mathbf{x}}\mathbf{s}_\theta(\mathbf{x})$ denotes the Jacobian of $\mathbf{s}_\theta(\mathbf{x})$ with respect to $\mathbf{x}$.

The proof is straightforward. See proof of **Theorem 1** in [Hyvärinen and Dayan, 2005].

This is still intractable due to the high computational complexity of matrix trace.

Sliced Score Matching

- **Idea:** Replace full score vector comparison with 1D projections using random directions $\mathbf{v} \sim p_{\mathbf{v}}$ (a simple distribution, e.g., multivariate standard normal)
- **Objective:**

$$\frac{1}{2}\,\mathbb{E}_{p_{\mathbf{v}}}\,\mathbb{E}_{p_{\text{data}}}\left[\left(\mathbf{v}^{\top}\nabla_{\mathbf{x}}\log p_{\text{data}}(\mathbf{x}) - \mathbf{v}^{\top}\mathbf{s}_{\theta}(\mathbf{x})\right)^{2}\right]$$

  which is equivalent to:

$$\mathbb{E}_{p_{\mathbf{v}}}\,\mathbb{E}_{p_{\text{data}}}\left[\mathbf{v}^{\top}\nabla_{\mathbf{x}}\mathbf{s}_{\theta}(\mathbf{x})\,\mathbf{v} + \frac{1}{2}\left(\mathbf{v}^{\top}\mathbf{s}_{\theta}(\mathbf{x})\right)^{2}\right]$$

- **Implementation:** Hessian-vector products can be computed in $\mathcal{O}(1)$ backprop steps.
- **Finite-sample estimator:**

$$\frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[\mathbf{v}_{ij}^{\top}\nabla_{\mathbf{x}}\mathbf{s}_{\theta}(\mathbf{x}_i)\mathbf{v}_{ij} + \frac{1}{2}\left(\mathbf{v}_{ij}^{\top}\mathbf{s}_{\theta}(\mathbf{x}_i)\right)^{2}\right]$$

Denoising Score Matching

- **Idea:** Perturb data point $\mathbf{x}$ with noise to obtain $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ (tractable) and match scores under the perturbed distribution:

$$q_\sigma(\tilde{\mathbf{x}}) = \int q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\, p_{\text{data}}(\mathbf{x})\, d\mathbf{x}$$

- **Explicit Score Matching objective:**

$$J_{ESM_{q_\sigma}}(\theta) = \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\left\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})\right\|^2\right]$$

Denoising Score Matching

- **Idea:** Perturb data point $\mathbf{x}$ with noise to obtain $\tilde{\mathbf{x}} \sim q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$ (tractable) and match scores under the perturbed distribution:

$$q_\sigma(\tilde{\mathbf{x}}) = \int q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\, p_{\text{data}}(\mathbf{x})\, d\mathbf{x}$$

- **Explicit Score Matching objective:**

$$J_{ESM_{q_\sigma}}(\theta) = \frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}})}\left[\left\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})\right\|^2\right]$$

- **Denoising Score Matching objective:**

$$J_{DSM_{q_\sigma}}(\theta) = \frac{1}{2}\,\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})\, p_{\text{data}}(\mathbf{x})}\left[\left\|\mathbf{s}_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\right\|_2^2\right]$$

  $J_{ESM_{q_\sigma}}(\theta)$ and $J_{DSM_{q_\sigma}}(\theta)$ are equivalent. Proof in **Appendix** of [Vincent, 2011].

- **Benefit:** The model $\mathbf{s}_\theta(\tilde{\mathbf{x}})$ approximates score of the **perturbed distribution**, which naturally suits our setting.

Table of Contents

Sampling with Langevin Dynamics

- **Goal:** Generate samples from the target distribution $p(\mathbf{x})$ using only the score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$.
- **Langevin dynamics update:**
  Given a fixed step size $\epsilon > 0$, and an initial value $\tilde{\mathbf{x}}_0 \sim \pi(\mathbf{x})$ with π being a prior distribution, the Langevin update rule is:

$$\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2} \nabla_{\mathbf{x}} \log p(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\, \mathbf{z}_t, \quad \mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$$

  When $\epsilon \to 0$ and $T \to \infty$, the final sample $\tilde{\mathbf{x}}_T \sim p(\mathbf{x})$.
- **Key insight:** To sample from $p(\mathbf{x})$, we:
  - First train a score network such that

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

  - Then plug $\mathbf{s}_\theta(\mathbf{x})$ in the Langevin dynamics update to approximately sample from $p(\mathbf{x})$.

Problems of Langevin Dynamics

- **Manifold hypothesis issues:**
  - Real-world data often lie on a low-dimensional manifold embedded in high-dimensional space.
  - Score matching fails when the data support is not the full space.

Problems of Langevin Dynamics

- **Manifold hypothesis issues:**
  - Real-world data often lie on a low-dimensional manifold embedded in high-dimensional space.
  - Score matching fails when the data support is not the full space.
- **Low density regions:**
  - Data are sparse in low-density areas, making estimation of scores unreliable.
  - Low efficiency when crossing low-density regions between two modes.

SMLD Algorithm: Noise Conditional Score Networks

- **Noise schedule:** Define a geometric sequence of noise levels $\{\sigma_i\}_{i=1}^L$ that satisfies $\frac{\sigma_1}{\sigma_2} = \cdots = \frac{\sigma_{L-1}}{\sigma_L} > 1$, and the perturbed distribution:

$$q_{\sigma_i}(\tilde{\mathbf{x}}) = \int p_{\text{data}}(\mathbf{x}) \, \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma_i^2 \mathbf{I}) \, d\mathbf{x}$$

- **Denoising score matching:** Train a noise-conditional score model $\mathbf{s}_\theta(\mathbf{x}, \sigma_i) \approx \nabla_{\mathbf{x}} \log q_{\sigma_i}(\mathbf{x})$.
- **Objective:**

$$\ell(\theta; \sigma) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})} \left\| \mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2} \right\|_2^2$$

- **Unified Objective:**

$$\mathcal{L}(\theta; \{\sigma_i\}_{i=1}^L) = \frac{1}{L} \sum_{i=1}^{L} \lambda(\sigma_i) \, \ell(\theta; \sigma_i)$$

where $\lambda(\sigma_i) = \sigma_i^2$ is a weighting coefficient.

SMLD Algorithm: Annealed Langevin Dynamics

- **Goal:** Sample from $p_{\text{data}}(\mathbf{x}) \approx q_{\sigma_L}(\mathbf{x})$ using annealed Langevin dynamics across noise levels.
- **Procedure:**
  1. Start from prior $\tilde{\mathbf{x}}_0 \sim \mathcal{U}$ or $\mathcal{N}(0, I)$
  2. For $i = 1 \to L$:
     - Set step size $\alpha_i = \epsilon \cdot \sigma_i^2 / \sigma_L^2$
     - Run Langevin dynamics with score $\mathbf{s}_\theta(\mathbf{x}, \sigma_i)$
     - Initialize next level with the final sample of this one
  3. Final step targets $\sigma_L \to 0 \Rightarrow q_{\sigma_L}(\mathbf{x}) \approx p_{\text{data}}(\mathbf{x})$
- **Benefits:** Avoids manifold hypothesis issues by smoothing data and gradually refining score accuracy.

SMLD Performs Score Based Generative Modeling

Observation 1

SMLD aligns with the general scheme of score-based generative modeling.

SMLD: Forward Process (Variance Exploding SDE)

Given noise levels $\{\sigma_i\}_{i=1}^N$ with $\sigma_0 = 0, \sigma_1 < \sigma_2 < \cdots < \sigma_N$, each perturbation kernel can be derived from the following Markov chain:

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} \cdot \mathbf{z}_{i-1}, \quad \mathbf{z}_{i-1} \sim \mathcal{N}(0, \mathbf{I})$$

Let $\Delta t = \frac{1}{N}$, and let $\sigma(t)$ be a continuous interpolation of σ_i, then:

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \sqrt{\sigma^2(t + \Delta t) - \sigma^2(t)} \cdot \mathbf{z}(t)$$

Use definition of derivative:

$$\sqrt{\sigma^2(t + \Delta t) - \sigma^2(t)} \approx \sqrt{\frac{d[\sigma^2(t)]}{dt} \cdot \Delta t}$$

So we can approximate:

$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot \sqrt{\Delta t} \cdot \mathbf{z}(t)$$

Therefore, we obtain the **forward SDE** of SMLD:

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w}$$

SMLD: Reverse Process

- **Reverse SDE:**

$$d\bar{\mathbf{x}} = -\frac{d[\sigma^2(t)]}{dt} \cdot \nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{x}})\, dt + \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\bar{\mathbf{w}}$$

- **Discretization:**

$$\bar{\mathbf{x}}_i^m = \bar{\mathbf{x}}_i^{m-1} + \frac{\epsilon_i}{2} \cdot \nabla_{\mathbf{x}} \log p_{\sigma_i}(\bar{\mathbf{x}}_i^{m-1}) + \sqrt{\epsilon_i} \cdot \mathbf{z}_i^m, \quad \mathbf{z}_i \sim \mathcal{N}(0, \mathbf{I}), m = 1, 2, \cdots, M$$

  where $\epsilon_i \propto \sigma_i^2$, and p_{σ_i} is the noisy distribution at level σ_i

- **Score estimation:** We approximate $\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$ by

$$\mathbf{s}_\theta(\mathbf{x}, \sigma_i) \approx \nabla_{\mathbf{x}} \log q_{\sigma_i}(\mathbf{x}) = \mathbb{E}_{p_{\text{data}}(\mathbf{x})} \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma_i^2 \mathbf{I})$$

- **Denoising Score Matching loss:**

$$\ell(\theta; \sigma_i) = \frac{1}{2} \mathbb{E}_{\tilde{\mathbf{x}}, \mathbf{x}} \left\| \mathbf{s}_\theta(\tilde{\mathbf{x}}, \sigma_i) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma_i^2} \right\|^2$$

- **Interpretation:** SMLD trains $\mathbf{s}_\theta(\mathbf{x}, \sigma_i) \approx \nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x})$, and uses it to discretely simulate reverse SDE via Langevin dynamics.

Table of Contents

DDPM Algorithm

Algorithm 1 Training

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  Take gradient descent step on
  $$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Figure 2: DDPM Algorithm. Source: [Ho et al., 2020]

DDPM Performs Score Based Generative Modeling

Observation 2

DDPM aligns with the general scheme of score-based generative modeling.

DDPM: Forward Process (Variance Perserving SDE)

- **Forward SDE:**

$$dx = -\frac{1}{2}\beta(t)\mathbf{x}\,dt + \sqrt{\beta(t)}\,d\mathbf{w}$$

where $\beta(t)$ is the continuous version of the noise schedule $\{\beta_t\}_{t=1}^{T}$.

- **Discrete formulation:** Use $\{\beta_t\}_{t=1}^{T}$ to define:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

- **Closed-form marginal:**

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}), \quad \bar{\alpha}_t := \prod_{i=1}^{t}(1-\beta_i)$$

- **Interpretation:** Forward process gradually corrupts $\mathbf{x}_0 \sim p_{\text{data}}$ into Gaussian noise.

DDPM: Reverse Process

- **Reverse SDE:**

$$d\bar{\mathbf{x}} = \left(-\frac{1}{2}\beta(t)\bar{\mathbf{x}} - \beta(t)\nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{x}})\right) dt + \sqrt{\beta(t)}\, d\bar{\mathbf{w}}$$

- **Score network:**

$$\mathbf{s}_\theta(\bar{\mathbf{x}}, t) \approx \nabla_{\mathbf{x}} \log p_t(\bar{\mathbf{x}})$$

- **Plug in reverse SDE:**

$$d\bar{\mathbf{x}} = \left(-\frac{1}{2}\beta(t)\bar{\mathbf{x}} - \beta(t)\mathbf{s}_\theta(\bar{\mathbf{x}}, t)\right) dt + \sqrt{\beta(t)}\, d\bar{\mathbf{w}}$$

DDPM: Discretization

- Given time grid $t = T, T-1, \ldots, 1$, we have the reverse sampling update:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \left(\frac{1}{2}\beta_t\mathbf{x}_t + \beta_t\mathbf{s}_\theta(\mathbf{x}_t, t)\right)\cdot\Delta t + \sqrt{\beta_t\cdot\Delta t}\cdot\mathbf{z}_t \quad \mathbf{z}_t \sim \mathcal{N}(0, \mathbf{I})$$

- $\Delta t = 1$, simplifying to:

$$\mathbf{x}_{t-1} = \mathbf{x}_t + \frac{1}{2}\beta_t\mathbf{x}_t + \beta_t\mathbf{s}_\theta(\mathbf{x}_t, t) + \sqrt{\beta_t}\cdot\mathbf{z}_t$$

- Plug in $\mathbf{s}_\theta(\mathbf{x}_t, t) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ and the approximation $1 + \frac{\beta_t}{2} \approx \frac{1}{\sqrt{1-\beta_t}} = \frac{1}{\sqrt{\alpha_t}}$, and define $\sigma_t = \sqrt{\beta_t}$ we have

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\sqrt{\alpha_t}(1-\alpha_t)}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sigma_t\mathbf{z}_t$$

which approximates the DDPM sampling procedure since $\beta_t \ll 1$.

Noise Prediction as Score Matching

- **Recall marginal distribution of forward process:**

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

- **Compute score:**

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0) = -\frac{1}{1-\bar{\alpha}_t}\left(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0\right)$$

- **Reparameterize $\mathbf{x}_t$ as:**

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

- **Plug into the score expression:**

$$\nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0) = -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}$$

  e true score at $\mathbf{x}_t$ is proportional to the negative noise!

- **Training objective:**

$$\min_\theta \mathbb{E}_{\mathbf{x}_0, t, \boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right\|^2\right] \quad \Rightarrow \quad \boldsymbol{\epsilon}_\theta \approx \boldsymbol{\epsilon}$$

- **Hence:**

$$\boxed{\mathbf{s}_\theta(\mathbf{x}_t, t) := -\frac{1}{\sqrt{1-\bar{\alpha}_t}}\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t \mid \mathbf{x}_0)}$$

Table of Contents

Change of Variables

Theorem 2 (Change of variables in one dimension)

Let X be a continuous random variable with PDF f_X, and let $Y = g(X)$, where g is differentiable and monotone. Then the PDF of Y is given by:

$$f_Y(y) = f_X(x) \left| \frac{dx}{dy} \right|, \quad \text{where} \quad x = g^{-1}(y).$$

The support of Y is all $g(x)$ with x in the support of X.

Change of Variables (cont'd)

Theorem 3 (Change of variables in multiple dimensions)

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a continuous random vector with joint PDF $f_{\mathbf{X}}$. Let g be an invertible function, $\mathbf{Y} = g(\mathbf{X})$, and mirror this by letting $\mathbf{y} = g(\mathbf{x})$. Since g is invertible, we also have $\mathbf{X} = g^{-1}(\mathbf{Y})$ and $\mathbf{x} = g^{-1}(\mathbf{y})$.

Suppose that all the partial derivatives $\frac{\partial x_i}{\partial y_j}$ exist and are continuous, so we can form the Jacobian matrix:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \cdots & \frac{\partial x_1}{\partial y_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_n}{\partial y_1} & \cdots & \frac{\partial x_n}{\partial y_n} \end{pmatrix}.$$

Also, assume that the determinant of this Jacobian matrix is non-zero. Then the joint PDF of $\mathbf{Y}$ is:

$$f_{\mathbf{Y}}(\mathbf{y}) = f_{\mathbf{X}}(g^{-1}(\mathbf{y})) \cdot | \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right| |$$

The inner bars around the Jacobian indicate taking the determinant, and the outer bars indicate taking the absolute value.

Model Architecture

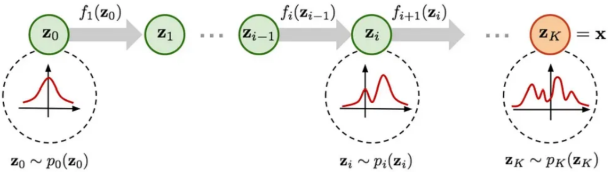

Figure 3: Model Architecture. Source: [Zhou,]

Likelihood Calculation

- Use a series of invertible transformations to map $\mathbf{z_0}$ to data $\mathbf{x}$:

$$p_i(\mathbf{z}_i) = p_{i-1}\left(f_i^{-1}(\mathbf{z}_i)\right)\left|\det J_{f_i^{-1}}\right| = p_{i-1}\left(\mathbf{z}_{i-1}\right)\left|\det J_{f_i}\right|^{-1}$$

  So we have the likelihood:

$$\begin{aligned}\log p(\mathbf{x}) &= \log p_0\left(f^{-1}(\mathbf{x})\right) + \sum_{i=1}^{K} \log\left|\det J_{f_i^{-1}}\right| \\ &= \log p_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log\left|\det J_{f_i}\right|\end{aligned}$$

- Parameterize the transformations using neural networks:

$$\begin{aligned}\log p_\theta(\mathbf{x}) &= \log p_0\left(f_\theta^{-1}(\mathbf{x})\right) + \sum_{i=1}^{K} \log\left|\det J_{f_{\theta_i}^{-1}}\right| \\ &= \log p_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log\left|\det J_{f_{\theta_i}}\right|\end{aligned}$$

Loss Calculation

- **Case 1: Samples available, form of $p(\mathbf{x})$ unknown**
  - Maximum likelihood estimation:

$$\theta^* = \arg\max_\theta \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(x^{(i)}) = \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N} \left(-\log p_\theta(x^{(i)})\right)$$

  - Training is actually learning the inverse transformation $f_\theta^{-1}(\mathbf{x})$, since we need to flow from $\mathbf{x}$ to $\mathbf{z}_0$ to calculate likelihood.
- **Case 2: Samples unavailable, form of $p(\mathbf{x})$ known**:
  - Reverse KL divergence:

$$\theta^* = \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N} \left[\log p_\theta(f_\theta(\mathbf{z}_0)) - \log p(f_\theta(\mathbf{z}_0))\right], \quad \mathbf{z}_0 \sim p_0(\mathbf{z}_0)$$

  - Training is actually learning the transformation $f_\theta(\mathbf{z_0})$.
  - A remarkable example: the **Boltzmann generator** from [Noé et al., 2019], one of the first works that leverage deep learning for unbiased, one-shot equilibrium sampling of Boltzmann distribution.

Flow Construction: Real NVP (Real-valued Non-Volume Preserving)

- **A flow of invertible transformations:**

$$\begin{cases} \mathbf{y}_{1:d} = \mathbf{x}_{1:d} \\ \mathbf{y}_{d+1:D} = \mathbf{x}_{d+1:D} \odot \exp(s(\mathbf{x}_{1:d})) + t(\mathbf{x}_{1:d}) \end{cases}$$

- **Invertible:**

$$\Leftrightarrow \begin{cases} \mathbf{x}_{1:d} = \mathbf{y}_{1:d} \\ \mathbf{x}_{d+1:D} = (\mathbf{y}_{d+1:D} - t(\mathbf{y}_{1:d})) \odot \exp(-s(\mathbf{y}_{1:d})) \end{cases}$$

- **Triangular Jacobian:**

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}^T} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial \mathbf{y}_{d+1:D}}{\partial \mathbf{x}_{1:d}^T} & \operatorname{diag}\left(\exp(s(\mathbf{x}_{1:d}))\right) \end{bmatrix}$$

$$\det(J) = \prod_{j=1}^{D-d} \exp(s(\mathbf{x}_{1:d})_j) = \exp\left(\sum_{j=1}^{D-d} s(\mathbf{x}_{1:d})_j\right)$$

Table of Contents

Motivation: Continuous Normalizing Flows (CNFs)

- Flow construction: full-rank residual flows, **discrete** case:

$$\phi_k(x) = x + \delta u_k(x), \phi = \phi_K \circ \cdots \circ \phi_2 \circ \phi_1$$

- Rearrange:

$$\frac{\phi(x) - x}{\delta} = u(x)$$

- Take continuous-time limit ($\delta \to 0$):

$$\frac{dx_t}{dt} = \lim_{\delta \to 0} \frac{x_{t+\delta} - x_t}{\delta} = \frac{\phi_t(x_t) - x_t}{\delta} = u_t(x_t)$$

- The **continuous** flow $\phi_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined by:

$$\frac{d\phi_t(x_0)}{dt} = u_t(\phi_t(x_0))$$

- Thus, ϕ_t maps initial condition x_0 to the ODE solution at time t:

$$x_t \triangleq \phi_t(x_0) = x_0 + \int_0^t u_s(x_s)\, ds$$

CNFs: Likelihood Computation

- Apply FP equation to compute the change in log-density:

$$\frac{\partial}{\partial t} p_t(x_t) = -\left(\nabla \cdot (u_t p_t)\right)(x_t).$$

- Take total derivative:

$$\frac{d}{dt} \log p_t(x_t) = -(\nabla \cdot u_t)(x_t)$$

- Solution:

$$\log p_t(x) = \log p_0(x_0) - \int_0^t (\nabla \cdot u_s)(x_s) ds$$

- Parameterize the vector field with neural network $v_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$, whose parameter denoted as θ:

$$\log p_\theta(x) \triangleq \log p_1(x) = \log p_0(x_0) - \int_0^1 \left(\nabla \cdot v_t\right)(x_t)\, dt.$$

CNFs: Expensive Training

- Train by maximizing expected log-likelihood of terminal samples:

$$\mathcal{L}(\theta) = \mathbb{E}_{x \sim q_1}[\log p_1(x)]$$

  where $q_1(x)$ is the distribution of data samples.
- **Challenges:**
  - Expensive numerical ODE simulations.
  - Requires estimators of divergence that scale well in high dimensions.
- Need alternative methods!

Flow Matching

- **Goal:** Construct a flow (a series of variables) which starts from simple prior p_0 and approximately ends at target distribution $q(x_1)$.
- There are many such paths, we just need to construct one.
- Then we can generate samples by sampling from p_0 and let them evolve according to the path.

Flow Matching

- **Goal:** Construct a flow (a series of variables) which starts from simple prior p_0 and approximately ends at target distribution $q(x_1)$.
- There are many such paths, we just need to construct one.
- Then we can generate samples by sampling from p_0 and let them evolve according to the path.
- Naive **Flow Matching objective:**

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t,\, p_t(x)} \left\| v_t(x) - u_t(x) \right\|^2$$

  where $p_t(x)$ is the target probability density path, and a corresponding vector field $u_t(x)$ which generates $p_t(x)$. $t \sim \mathcal{U}[0, 1]$.
- **Intractable:** p_t and u_t are unknown.

Conditional Flow Matching

- **Marginal probability path:**
  - For a data sample x_1, define the *conditional probability path* $p_t(x|x_1)$ such that $p_0(x|x_1) = p_0(x)$ and $p_1(x|x_1)$ is centered closely around $x = x_1$.
  - Marginalizing over $q(x_1)$ yields the marginal path:

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1$$

  - $p_t(x)$ **exactly satisfies our goal!**
- **Marginal vector field:**
  - By marginalizing the conditional vector fields $u_t(\cdot|x_1)$, we define the marginal vector field:

$$u_t(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)q(x_1)}{p_t(x)}dx_1$$

    where $u_t(\cdot|x_1) : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is a conditional vector field that generates $p_t(\cdot|x_1)$.
  - **The marginal vector field generates the marginal probability path**.

Conditional Flow Matching (cont'd)

Theorem 4 (Marginal vector field generates marginal path)

Given vector fields $u_t(x|x_1)$ that generate conditional probability paths $p_t(x|x_1)$, for any distribution $q(x_1)$, the marginal vector field u_t generates the marginal probability path p_t.

Proof

To verify this, we check that p_t and u_t satisfy the FP equation:

$$
\begin{aligned}
\frac{d}{dt} p_t(x) &= \int \left(\frac{d}{dt} p_t(x|x_1) \right) q(x_1) dx_1 \\
&= -\int \operatorname{div} \left(u_t(x|x_1) p_t(x|x_1) \right) q(x_1) dx_1 \\
&= -\operatorname{div} \left(\int u_t(x|x_1) p_t(x|x_1) q(x_1) dx_1 \right) \\
&= -\operatorname{div} \left(u_t(x) p_t(x) \right),
\end{aligned}
$$

The first and third equalities are justified by assuming the integrands satisfy the regularity conditions of the Leibniz Rule (for exchanging integration and differentiation).

Tractable Objective

- **Conditional Flow Matching (CFM) objective:**

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t,\, q(x_1),\, p_t(x|x_1)} \left\| v_t(x) - u_t(x|x_1) \right\|^2$$

  where $t \sim \mathcal{U}[0,1]$, $x_1 \sim q(x_1)$, and $x \sim p_t(x|x_1)$.

- **The FM and CFM objectives have identical gradients w.r.t. θ.**
- Consequently, we can train a CNF to generate the **marginal** probability path p_t. All we need are suitable **conditional** probability paths and vector fields.

Tractable Objective (cont'd)

Theorem 5 ($\mathcal{L}_{\mathrm{FM}} \equiv \mathcal{L}_{\mathrm{CFM}}$)

Assuming that $p_t(x) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0, 1]$, then, up to a constant independent of θ, $\mathcal{L}_{\mathrm{CFM}}$ and $\mathcal{L}_{\mathrm{FM}}$ are equal. Hence,

$$\nabla_\theta \mathcal{L}_{\mathrm{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\mathrm{CFM}}(\theta).$$

Proof

To ensure existence of all integrals and to allow the changing of integration order (by Fubini's Theorem) in the following we assume that $q(x)$ and $p_t(x|x_1)$ are decreasing to zero at a sufficient speed as $\|x\| \to \infty$, and that u_t, v_t, $\nabla_\theta v_t$ are bounded.
First, expand the squares:

$$\|v_t(x) - u_t(x)\|^2 = \|v_t(x)\|^2 - 2\langle v_t(x), u_t(x)\rangle + \|u_t(x)\|^2$$

$$\|v_t(x) - u_t(x|x_1)\|^2 = \|v_t(x)\|^2 - 2\langle v_t(x), u_t(x|x_1)\rangle + \|u_t(x|x_1)\|^2$$

Next, since u_t **is independent of** θ and note that

$$\mathbb{E}_{p_t(x)}\|v_t(x)\|^2 = \int \|v_t(x)\|^2 p_t(x)dx = \int\int \|v_t(x)\|^2 p_t(x|x_1)q(x_1)dx_1dx$$

$$= \mathbb{E}_{q(x_1),p_t(x|x_1)}\|v_t(x)\|^2,$$

Next,

$$\begin{aligned}
\mathbb{E}_{p_t(x)}\langle v_t(x), u_t(x)\rangle &= \int \left\langle v_t(x), \frac{\int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1}{p_t(x)} \right\rangle p_t(x)dx \\
&= \int \left\langle v_t(x), \int u_t(x|x_1)p_t(x|x_1)q(x_1)dx_1 \right\rangle dx \\
&= \int\int \langle v_t(x), u_t(x|x_1)\rangle p_t(x|x_1)q(x_1)dx_1dx \\
&= \mathbb{E}_{q(x_1),p_t(x|x_1)}\langle v_t(x), u_t(x|x_1)\rangle
\end{aligned}$$

Conditional Probability Paths and Vector Fields

- **The Conditional Flow Matching objective works with any choice of conditional probability path and conditional vector fields.**
- Consider the construction of $p_t(x|x_1)$ and $u_t(x|x_1)$ for Gaussian conditional probability paths:

$$p_t(x|x_1) = \mathcal{N}(x \mid \mu_t(x_1), \sigma_t(x_1)^2 I)$$

  where

  - $\mu : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is the time-dependent mean,
  - $\sigma : [0,1] \times \mathbb{R} \to \mathbb{R}_{>0}$ is the time-dependent std,
  - $\mu_0(x_1) = 0$, $\sigma_0(x_1) = 1$, so $p_0(x \mid x_1) = \mathcal{N}(x \mid 0, I)$
  - $\mu_1(x_1) = x_1$, $\sigma_1(x_1) = \sigma_{\min}$, which is set sufficently small, so $p_1(x \mid x_1)$ is a Gaussian dist. centered closely at x_1.
  - μ and σ are set, not learned.

Conditional Probability Paths and Vector Fields (cont'd)

- Consider the flow conditioned on x_1:

$$\psi_t(x) = \sigma_t(x_1)x + \mu_t(x_1)$$

  where x is distributed as a standard Gaussian.

- This flow yields a vector field that generates the conditional probability path:

$$\frac{d}{dt}\psi_t(x) = u_t(\psi_t(x)|x_1)$$

- Reparameterizing $p_t(x|x_1)$ in terms of just x_0 and substituting into the CFM loss:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p(x_0)} \left\| v_t(\psi_t(x_0)) - \frac{d}{dt}\psi_t(x_0) \right\|^2$$

- Since ψ_t is invertible, we can also solve for u_t in closed form.

Closed Form Calculation

Theorem 6 (Closed form of u_t)

Let $p_t(x|x_1)$ be a Gaussian probability path as defined earlier, and ψ_t its corresponding flow map. Then, the unique vector field that defines ψ_t has the form:

$$u_t(x|x_1) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)} (x - \mu_t(x_1)) + \mu_t'(x_1).$$

where $f' = \frac{d}{dt} f$, for a time-dependent function f.
Consequently, $u_t(x|x_1)$ generates the Gaussian path $p_t(x|x_1)$.

Proof

For notational simplicity let $w_t(x) = u_t(x \mid x_1)$. We have:

$$\frac{d}{dt}\psi_t(x) = w_t(\psi_t(x)).$$

Since ψ_t is invertible, we let $x = \psi_t^{-1}(y)$ and get

$$\psi_t'(\psi_t^{-1}(y)) = w_t(y).$$

Now, inverting $\psi_t(x)$ provides

$$\psi_t^{-1}(y) = \frac{y - \mu_t(x_1)}{\sigma_t(x_1)}.$$

Differentiating ψ_t with respect to t gives

$$\psi_t'(x) = \sigma_t'(x_1)x + \mu_t'(x_1).$$

Plugging back the last two equations we get

$$w_t(y) = \frac{\sigma_t'(x_1)}{\sigma_t(x_1)}\left(y - \mu_t(x_1)\right) + \mu_t'(x_1)$$

as required.

Summary

- Score-based model: Add noise first, then learn the "denoising" direction (score).
- Flow-based model: Learn an invertible transformation to convert noise into data.
- Flow matching: Learn the probability flow along arbitrary paths.
- Rectified flow: Learn the probability flow along straight-line paths (the simplest case of flow matching).

References I

Blitzstein, J. K. and Hwang, J. (2019).
Introduction to probability.
Chapman and Hall/CRC.

Fjelde, T., Mathieu, E., and Dutordoir, V. (2024).
An introduction to flow matching.
`https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html`.
Machine Learning Group, Department of Engineering, University of Cambridge.

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

References II

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

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. (2022).
Flow matching for generative modeling.
arXiv preprint arXiv:2210.02747.

Noé, F., Olsson, S., Köhler, J., and Wu, H. (2019).
Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning.
Science, 365(6457):eaaw1147.

References III

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

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020).
Score-based generative modeling through stochastic differential equations.
arXiv preprint arXiv:2011.13456.

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

References IV

Zhou, K.
Flow-based generative model intro.
Lecture 13, FIAS Deep Learning Course.

Thank you!

Any questions?