COMP 7070 Advanced Topics in AI and ML April 16, 2024

Lecture 13: Generative Models

Instructor: Yifan Chen *Scribes: Xiong PENG* *Proof reader: Zhanke Zhou*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

13.1 DDIM

13.1.1 Background

Given samples from a data distribution $q(x_0)$, we are interested in learning a model distribution $p_\theta(x_0)$ that approximates $q(x_0)$ and is easy to sample from. Denoising diffusion probabilistic models (DDPMs) are latent variable models of the form:

$$p_\theta(x_0) = \int p_\theta(x_{0:T})\, dx_{1:T}, \quad \text{where } p_\theta(x_{0:T}) := p_\theta(x_T) \prod_{t=1}^{T} p_\theta^{(t)}(x_{t-1} \mid x_t), \tag{13.1}$$

where $x_1, \ldots, x_T$ are latent variables in the same sample space as x_0 (denoted as $\mathcal{X}$). The parameters θ are learned to fit the data distribution $q(x_0)$ by maximizing a variational lower bound:

$$\max_\theta \mathbb{E}_{q(x_0)}[\log p_\theta(x_0)] \le \max_\theta \mathbb{E}_{q(x_{0:T})}\left[\log p_\theta(x_{0:T}) - \log q(x_{1:T} \mid x_0)\right], \tag{13.2}$$

where $q(x_{1:T} \mid x_0)$ is some inference distribution over the latent variables. Unlike typical latent variable models (such as the variational autoencoder), DDPMs are learned with a fixed (rather than trainable) inference procedure $q(x_{1:T} \mid x_0)$, and latent variables are relatively high dimensional. For example, DDPM considered the following Markov chain with Gaussian transitions parameterized by a decreasing sequence $\alpha_{1:T} \in (0,1]^T$:

$$q(x_{1:T} \mid x_0) := \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \quad \text{where } q(x_t \mid x_{t-1}) := \mathcal{N}\left(\sqrt{\frac{\alpha_t}{\alpha_{t-1}}} x_{t-1}, \left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right) I\right), \tag{13.3}$$

where the covariance matrix is ensured to have positive terms on its diagonal. This is called the forward process due to the autoregressive nature of the sampling procedure (from x_0 to x_T). We call the latent variable model $p_\theta(x_{0:T})$, which is a Markov chain that samples from x_T to x_0, the generative process, since it approximates the intractable reverse process $q(x_{1:T} \mid x_0)$. Intuitively, the forward process progressively adds noise to the observation x_0, whereas the generative process progressively denoises a noisy observation.

A special property of the forward process is that:

$$q(x_t \mid x_0) := \int q(x_t \mid x_{t-1}) q(x_{t-1} \mid x_0)\, dx_{t-1} = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_0, (1 - \alpha_t) I\right). \tag{13.4}$$

When we set α_T sufficiently close to 0, $q(x_T \mid x_0)$ converges to a standard Gaussian for all x_0, so it is natural to set $p_\theta(x_T) := \mathcal{N}(0, I)$. If all the conditionals are modeled as Gaussians with trainable mean functions and fixed variances, the objective in Eq. (13.2) can be simplified to:

$$L_\gamma(\epsilon_\theta) := \sum_{t=1}^{T} \gamma_t \mathbb{E}_{x_0 \sim q(x_0), \epsilon_t \sim \mathcal{N}(0,I)} \left[\left\| \epsilon_\theta^{(t)}\left(\sqrt{\alpha_t} x_0 + \sqrt{1 - \alpha_t} \epsilon_t\right) - \epsilon_t \right\|_2^2 \right], \tag{13.5}$$

where $\epsilon_\theta^{(t)} := \epsilon_\theta^{(t)}(x_t)$ is a set of T functions, each $\epsilon_\theta^{(t)} : \mathcal{X} \to \mathcal{X}$ (indexed by t) is a function with trainable parameters $\theta^{(t)}$, and $\gamma := [\gamma_1, \ldots, \gamma_T]$ is a vector of positive coefficients in the objective that depends on $\alpha_{1:T}$.

13.1.2 Variational Inference for Non-Markovian Forward Processes

Because the generative model approximates the reverse of the inference process, we need to rethink the inference process in order to reduce the number of iterations required by the generative model. Our key observation is that the DDPM objective in the form of L_γ only depends on the marginals $q(x_t \mid x_0)$, but not directly on the joint $q(x_{1:T} \mid x_0)$. Since there are many inference distributions (joints) with the same marginals, we explore non-Markovian alternative inference processes, which leads to new generative processes. These non-Markovian inference processes lead to the same surrogate objective function as DDPMs, as we will show below.

Non-Markovian Forward Processes

Let us consider a family $\mathcal{Q}$ of inference distributions, indexed by a real vector $\sigma \in \mathbb{R}^T_{\geq 0}$:

$$q_\sigma(x_{1:T} \mid x_0) := q_\sigma(x_T \mid x_0) \prod_{t=2}^{T} q_\sigma(x_{t-1} \mid x_t, x_0), \tag{13.6}$$

where $q_\sigma(x_T \mid x_0) = \mathcal{N}(\sqrt{\alpha_T}x_0, (1-\alpha_T)I)$ and for all $t > 1$,

$$q_\sigma(x_{t-1} \mid x_t, x_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}x_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\frac{x_t - \sqrt{\alpha_t}x_0}{\sqrt{1-\alpha_t}}, \sigma_t^2 I\right). \tag{13.7}$$

The mean function is chosen in order to ensure that $q_\sigma(x_t \mid x_0) = \mathcal{N}(\sqrt{\alpha_t}x_0, (1-\alpha_t)I)$ for all t, so that it defines a joint inference distribution that matches the "marginals" as desired. The forward process can be derived from Bayes' rule:

$$q_\sigma(x_{t-1} \mid x_t, x_0) = \frac{q_\sigma(x_t \mid x_{t-1}, x_0) q_\sigma(x_{t-1} \mid x_0)}{q_\sigma(x_t \mid x_0)}, \tag{13.8}$$

which is also Gaussian (although we do not use this fact for the remainder of this paper). Unlike the diffusion process in Eq. (13.3), the forward process here is no longer Markovian, since each x_t could depend on both x_{t-1} and x_0. The magnitude of σ controls the degree of stochasticity of the forward process; when $\sigma \to 0$, we reach an extreme case where as long as we observe x_0 and x_t for some t, then x_{t-1} becomes known and fixed.

Generative Process and Unified Variational Inference Objective

Next, we define a trainable generative process $p_\theta(x_{0:T})$ where each $p_\theta^{(t)}(x_{t-1} \mid x_t)$ leverages knowledge of $q_\sigma(x_{t-1} \mid x_t, x_0)$. Intuitively, given a noisy observation x_t, we first predict the corresponding x_0, and then use it to obtain a sample x_{t-1} through the reverse conditional distribution $q_\sigma(x_{t-1} \mid x_t, x_0)$, which we have defined.

For some $x_0 \sim q(x_0)$ and $\epsilon_t \sim \mathcal{N}(0, I)$, x_t can be obtained using Eq. (13.4). The model $\epsilon_\theta^{(t)}(x_t)$ then attempts to predict ϵ_t from x_t, without knowledge of x_0. By rewriting Eq. (13.4), one can then predict the denoised observation, which is a prediction of x_0 given x_t:

$$f_\theta^{(t)}(x_t) := (x_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(x_t))/\sqrt{\alpha_t}. \tag{13.9}$$

We can then define the generative process with a fixed prior $p_\theta(x_T) = \mathcal{N}(0, I)$ and

$$p_\theta^{(t)}(x_{t-1} \mid x_t) = \begin{cases} \mathcal{N}(f_\theta^{(1)}(x_t), \sigma_t^2 I) & \text{if } t = 1 \\ q_\sigma(x_{t-1} \mid x_t, f_\theta^{(t)}(x_t)) & \text{otherwise,} \end{cases} \tag{13.10}$$

where $q_\sigma(x_{t-1} \mid x_t, f_\theta^{(t)}(x_t))$ is defined as in Eq. (13.7) with x_0 replaced by $f_\theta^{(t)}(x_t)$. We add some Gaussian noise (with covariance $\sigma_t^2 I$) for the case of $t = 1$ to ensure that the generative process is supported everywhere.

We optimize θ via the following variational inference objective (which is functional over ϵ_0):

$$J_\sigma(\epsilon_0) := \mathbb{E}_{x_0 \sim q(x_0)}[\log q_\sigma(x_{1:T} \mid x_0) - \log p_\theta(x_{0:T})] \tag{13.11}$$

$$= \mathbb{E}_{x_0 \sim q(x_0)} \left[\log q_\sigma(x_T \mid x_0) + \sum_{t=2}^{T} \log q_\sigma(x_{t-1} \mid x_t, x_0) - \sum_{t=1}^{T} \log p_\theta^{(t)}(x_{t-1} \mid x_t) - \log p_\theta(x_T) \right], \tag{13.12}$$

where we factorize $q_\sigma(x_{1:T} \mid x_0)$ according to Eq. (13.6) and $p_\theta(x_{0:T})$ according to Eq. (13.1).

13.1.3 Denoising Diffusion Implicit Models

From $p_\theta(x_{1:T})$ in Eq. (13.10), one can generate a sample x_{t-1} from a sample x_t via:

$$x_{t-1} = \sqrt{\alpha_{t-1}} \left(\frac{x_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta^{(t)}(x_t)}{\sqrt{\alpha_t}} \right) + \sqrt{1 - \alpha_{t-1} - \sigma_t^2} \cdot \epsilon_\theta^{(t)}(x_t) + \sigma_t \epsilon_t, \tag{13.13}$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ is standard Gaussian noise independent of x_t, and we define $\alpha_0 = 1$. Different choices of σ or values result in different generative processes, all while using the same model ϵ_0, so re-training the model is unnecessary. When $\sigma_t = \sqrt{(1-\alpha_{t-1})/(1-\alpha_t)}\sqrt{1-\alpha_t/\alpha_{t-1}}$ for all t, the forward process becomes Markovian, and the generative process becomes a DDPM.

We note another special case when $\sigma_t = 0$ for all t; the forward process becomes deterministic given x_{t-1} and x_0, except for $t = 1$; in the generative process, the coefficient before the random noise ϵ_t becomes zero. The resulting model becomes an implicit probabilistic model, where samples are generated from latent variables with a fixed procedure (from x_T to x_0). We name this the denoising diffusion implicit model (DDIM) because it is an implicit probabilistic model trained with the DDPM objective (despite the fact that the forward process is no longer a diffusion).

13.2 FLOW MATCHING

13.2.1 Preliminaries: Continuous Normalizing Flows

Let $\mathbb{R}^d$ denote the data space with data points $x = (x^1, \ldots, x^d) \in \mathbb{R}^d$. Two important objects we use in this paper are:

- the *probability density path* $p : [0,1] \times \mathbb{R}^d \to \mathbb{R}_{>0}$, which is a time-dependent probability density function, i.e., $\int p_t(x)\, dx = 1$, and
- a time-dependent vector field $v : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$.

A vector field v_t can be used to construct a time-dependent diffeomorphic map, called a *flow*, $\phi : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$, defined via the ordinary differential equation (ODE):

$$\frac{d}{dt}\phi_t(x) = v_t(\phi_t(x)) \quad \text{and} \quad \phi_0(x) = x. \tag{13.14}$$

A Continuous Normalizing Flow (CNF) can be used to reshape a simple prior density p_0 (e.g., pure noise) to a more complicated one, p_1, via the push-forward equation:

$$p_t = [\phi_t]_* p_0, \tag{13.15}$$

where the push-forward (or change of variables) operator $*$ is defined by:

$$[\phi_t]_* p_0(x) = p_0(\phi_t^{-1}(x)) \det \left[\frac{\partial \phi_t^{-1}}{\partial x} \right]. \tag{13.16}$$

A vector field v_t is said to *generate* a probability density path p_t if its flow ϕ_t satisfies equation (13.15).

13.2.2 Flow Matching

Let x_1 denote a random variable distributed according to some unknown data distribution $q(x_1)$. We assume we only have access to data samples from $q(x_1)$ but no access to the density function itself. Furthermore, we let p_t be a probability path such that $p_0 = p$ is a simple distribution, e.g., the standard normal distribution $p(x) = \mathcal{N}(x|0, I)$, and let p_1 be approximately equal in distribution to q. We will later discuss how to construct such a path. The Flow Matching objective is then designed to match this target probability path, which will allow us to flow from p_0 to p_1.

We define the Flow Matching (FM) objective as:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p_t(x)}\left[\|v_t(x) - u_t(x)\|^2\right], \tag{13.17}$$

where θ denotes the learnable parameters of the CNF vector field v_t, $t \sim U[0, 1]$ (uniform distribution), and $x \sim p_t(x)$. Simply put, the FM loss regresses the vector field u_t with a neural network v_t. Upon reaching zero loss, the learned CNF model will generate $p_t(x)$.

Flow Matching is a simple and attractive objective, but naively on its own, it is intractable to use in practice since we have no prior knowledge for what an appropriate p_t and u_t are.

13.2.3 Constructing p_t, u_t from Conditional Probability Paths and Vector Fields

A simple way to construct a target probability path is via a mixture of simpler probability paths. Given a particular data sample x_1, we denote by $p_t(x|x_1)$ a conditional probability path such that it satisfies $p_0(x|x_1) = p(x)$ at time $t = 0$, and we design $p_1(x|x_1)$ at $t = 1$ to be a distribution concentrated around $x = x_1$, e.g., $p_1(x|x_1) = \mathcal{N}(x|x_1, \sigma^2 I)$, a normal distribution with x_1 mean and a sufficiently small standard deviation or $\sigma > 0$.

Marginalizing the conditional probability paths over $q(x_1)$ gives rise to the marginal probability path:

$$p_t(x) = \int p_t(x|x_1)q(x_1)\, dx_1. \tag{13.18}$$

Interestingly, we can also define a marginal vector field by "marginalizing" over the conditional vector fields in the following sense (we assume $p_t(x) > 0$ for all t and x):

$$u_t(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)q(x_1)}{p_t(x)}\, dx_1. \tag{13.19}$$

Theorem 1. *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 in Eq.* (13.19) *generates the marginal probability path p_t in Eq.* (13.18)*.*

13.2.4 Conditional Flow Matching

Unfortunately, due to the intractable integrals in the definitions of the marginal probability path and vector field (Eqs (13.18) and (13.19)), it is still intractable to compute u_t, and consequently, intractable to naively compute an unbiased estimator of the original Flow Matching objective. Instead, we propose a simpler objective, which surprisingly will result in the same optima as the original objective. Specifically, we consider the Conditional Flow Matching (CFM) objective:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p_t(x|x_1)}\left[\|v_t(x) - u_t(x|x_1)\|^2\right], \tag{13.20}$$

where $t \sim U[0, 1]$, $x_1 \sim q(x_1)$, and now $x \sim p_t(x|x_1)$. Unlike the FM objective, the CFM objective allows us to easily sample unbiased estimates as long as we can efficiently sample from $p_t(x|x_1)$ and compute $u_t(x|x_1)$, both of which can be easily done as they are defined on a per-sample basis.

Our key observation is therefore:

> **The FM (Eq.** (13.17)**) and CFM (Eq.** (13.20)**) objectives have identical gradients w.r.t. θ.**

That is, optimizing the CFM objective is equivalent (in expectation) to optimizing the FM objective. Consequently, this allows us to train a CNF to generate the marginal probability path p_t—which in particular, approximates the unknown data distribution q at $t = 1$—without ever needing access to either the marginal probability path or the marginal vector field. We simply need to design suitable conditional probability paths and vector fields. We formalize this property in the following theorem.

Theorem 2. *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, $\mathcal{L}_{CFM}$ and $\mathcal{L}_{FM}$ are equal. Hence, $\nabla_\theta \mathcal{L}_{FM}(\theta) = \nabla_\theta \mathcal{L}_{CFM}(\theta)$.*

13.2.5 Gaussian Conditional Probability Paths and Vector Fields

The CFM objective works with any choice of conditional probability path and vector field $u_t(x|x_1)$. In this section, we discuss the construction of $p_t(x|x_1)$ and $u_t(x|x_1)$ with a particular focus on conditional probability paths. Namely, we consider conditional Gaussian paths:

$$p_t(x|x_1) = \mathcal{N}(x|\mu_t(x_1), \sigma_t(x_1)^2 I), \tag{13.21}$$

where $\mu_t(x_1)$ is the time-dependent mean of the Gaussian distribution, while $\sigma_t(x_1)$ is a time-dependent scalar standard deviation (std). We set $\mu_0(x_1) = 0$ and design conditional paths such that all conditional probability paths converge to the same standard Gaussian at $t = 0$, e.g., $\mathcal{N}(x|0, I)$. We then set $\mu_1(x_1) = x_1$ and $\sigma_1(x_1) = \sigma_{\min}$, which ensures that $p_1(x|x_1)$ is a concentrated Gaussian distribution centered at x_1.

Let $u_t(x|x_1)$ denote the vector field that generates the conditional probability path:

$$\frac{d}{dt}\phi_t(x) = u_t(\phi_t(x)|x_1). \tag{13.22}$$

Plugging equation (13.22) into the CFM loss we obtain:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,q(x_1),p_0(x)}\left[\|v_t(\psi_t(x_0)) - \frac{d}{dt}\psi_t(x_0)\|^2\right]. \tag{13.23}$$

Since ψ_t is a simple (invertible) affine map, we can use equation (13.22) to solve for u_t in a closed form.

13.3 Rectified flow

Given empirical observations of $X_0 \sim \pi_0, X_1 \sim \pi_1$, the rectified flow induced from (X_0, X_1) is an ordinary differentiable model (ODE) on time $t \in [0, 1]$,

$$dZ_t = v(Z_t, t)dt,$$

which converts Z_0 from π_0 to a Z_1 following π_1. The drift force $v : \mathbb{R}^d \to \mathbb{R}^d$ is set to drive the flow to follow the direction $(X_1 - X_0)$ of the linear path pointing from X_0 to X_1 as much as possible, by solving a simple least squares regression problem:

$$\min_v \int_0^1 \mathbb{E}\left[\|(X_1 - X_0) - v(X_t, t)\|^2\right] dt, \quad \text{with} \quad X_t = tX_1 + (1-t)X_0, \tag{13.24}$$

where X_t is the linear interpolation of X_0 and X_1. Naviely, X_t follows the ODE of $dX_t = (X_1 - X_0)dt$, which is non-causal (or anticipating) as the update of X_t requires the information of the final point X_1. By fitting the drift v with $X_1 - X_0$, the rectified flow *causalizes* the linear interpolation X_t paths, yielding an ODE flow that can be simulated without seeing the future.

In practice, we parameterize v with a neural network or other nonlinear models and solve Eq. (13.24) with any off-the-shelf stochastic optimizer, such as stochastic gradient descent, with empirical draws of (X_0, X_1). See the rectified flow algorithm.

Algorithm 1 After we get v, we solve the ODE starting from $Z_0 \sim \pi_0$ to transfer π_0 to π_1, backwardly starting from $Z_1 \sim \pi_1$ to transfer π_1 to π_0. Specifically, for backward sampling, we simply

solve $d\tilde{X}_t = -v(\tilde{X}_t, t)dt$ initialized from $X_0 \sim \pi_1$ and set $\tilde{X}_t = \tilde{X}_{1-t}$. The forward and backward sampling are equally favored by the training algorithm because the objective in Eq. (13.24) is *time-symmetric* in that it yields the equivalent problem if we exchange X_0 and X_1 and flip the sign of v.

Flows Avoid Crossing A key to understanding the method is the non-crossing property of flows: the different paths following a well-defined ODE $dZ_t = v(Z_t, t)dt$, whose solution exists and is unique, cannot cross each other at any time $t \in [0, 1]$. Specifically, there exists no location $z \in \mathbb{R}^d$ and time $t \in [0, 1]$ such that two paths go across z at time t along different directions, because otherwise the solution of the ODE would be non-unique. On the other hand, the paths of the interpolation process X_t may intersect with each other, which makes it non-causal. Hence, the rectified flow *rewires* the individual trajectories passing through the interpolation points to avoid crossing while tracing out the same density map as the linear interpolation paths due to the optimization of Eq. (13.24). We can view the linear interpolation X_t as building roads (or tunnels) to connect π_0 and π_1, and the rectified flow as traffic of particles passing through the roads in a myopic, memoryless, non-crossing way, which allows them to ignore the global path information of how X_0 and X_1 are paired and rebuild a more deterministic pairing of (Z_0, Z_1).

Rectified Flows Reduce Transport Costs

If Eq. (13.24) is solved exactly, the pair (Z_0, Z_1) of the rectified flow is guaranteed to be a valid coupling of π_0, π_1 (Theorem 3.3), that is, Z_1 follows π_1 if $Z_0 \sim \pi_0$. Moreover, (Z_0, Z_1) guarantees to yield no larger transport cost than the data pair (X_0, X_1) simultaneously for all convex cost functions c (Theorem 3.5). The data pair (X_0, X_1) can be an arbitrary coupling of π_0, π_1, typically independent (i.e., $(X_0, X_1) \sim \pi_0 \times \pi_1$) as dictated by the lack of meaningfully paired observations in practical problems. In comparison, the rectified coupling (Z_0, Z_1) has a deterministic dependency as it is constructed from an ODE model. Denote by $(Z_0, Z_1) = \texttt{Rectify}((X_0, X_1))$ the mapping from (X_0, X_1) to (Z_0, Z_1). Hence, `Rectify` converts an arbitrary coupling into a deterministic coupling with lower convex transport costs.

Straight Line Flows Yield Fast Simulation

Following Algorithm 1, denote by $Z = \texttt{RectFlow}((X_0, X_1))$ the rectified flow induced from (X_0, X_1). Applying this operator recursively yields a sequence of rectified flows $Z^{k+1} = \texttt{RectFlow}((Z_0^k, Z_1^k))$ with $(Z_0^0, Z_1^0) = (X_0, X_1)$, where Z^k is the k-th rectified flow, or simply k-rectified flow, induced from (X_0, X_1).

This *reflow* procedure not only decreases transport cost but also has the important effect of straightening out the rectified flows, that is, making its flow paths more straight. This is highly attractive computationally as flows with nearly straight paths incur small time-discretization error in numerical simulation. Indeed, perfectly straight paths can be simulated exactly with a single Euler step and is effectively a one-step model. This addresses the very bottleneck of high inference cost in existing continuous-time ODE/SDE models.

13.4 Consistency Model

13.4.1 Background

Consistency models are heavily inspired by the theory of continuous-time diffusion models . Diffusion models generate data by progressively perturbing data to noise via Gaussian perturbations, then creating samples from noise via sequential denoising steps. Let $p_{\text{data}}(x)$ denote the data distribution. Diffusion models start by diffusing $p_{\text{data}}(x)$ with a stochastic differential equation (SDE):

$$d\mathbf{x}_t = \mu(\mathbf{x}_t, t)\, dt + \sigma(t)\, d\mathbf{w}_t, \tag{13.25}$$

where $t \in [0, T]$, $T > 0$ is a fixed constant, $\mu(\cdot, \cdot)$ and $\sigma(\cdot)$ are the drift and diffusion coefficients respectively, and $\{\mathbf{w}_t\}_{t\in[0,T]}$ denotes the standard Brownian motion. We denote the distribution of

$\mathbf{x}_t$ as $p_t(x)$ and as a result $p_0(x) \equiv p_{\text{data}}(x)$. A remarkable property of this SDE is the existence of an ordinary differential equation (ODE), dubbed the Probability Flow (PF) ODE, whose solution trajectories sampled at t are distributed according to $p_t(x)$:

$$d\mathbf{x}_t = \left[\mu(\mathbf{x}_t, t) - \frac{1}{2}\sigma(t)^2 \nabla \log p_t(\mathbf{x}_t)\right] dt. \tag{13.26}$$

Here $\nabla \log p_t(x)$ is the score function of $p_t(x)$; hence diffusion models are also known as score-based generative models.

Typically, the SDE in Eq. (13.25) is designed such that $p_T(x)$ is close to a tractable Gaussian distribution $\pi(x)$. We hereafter adopt the settings where $\mu(x,t) = 0$ and $\sigma(t) = \sqrt{2t}$. In this case, we have $p_t(x) = p_{\text{data}}(x) \otimes \mathcal{N}(0, t^2 I)$, where $\otimes$ denotes the convolution operator, and $\pi(x) = \mathcal{N}(0, T^2 I)$. For sampling, we first train a score model $s_\phi(x,t) \approx \nabla \log p_t(x)$ via score matching, then plug it into Eq. (13.26) to obtain an empirical estimate of the PF ODE, which takes the form of

$$\frac{d\mathbf{x}_t}{dt} = -t s_\phi(\mathbf{x}_t, t). \tag{13.27}$$

We call Eq. (13.27) the *empirical PF ODE*. Next, we sample $\mathbf{x}_T \sim \pi = \mathcal{N}(0, T^2 I)$ to initialize the empirical PF ODE and solve it backwards in time with any numerical ODE solver such as Euler and Heun to obtain the solution trajectory $\{\mathbf{x}_t\}_{t\in[0,T]}$. The resulting $\mathbf{x}_0$ can then be viewed as an approximate sample from the data distribution $p_{\text{data}}(x)$. To avoid numerical instability, one typically stops the solver at $t = e$, which is a fixed small positive number, and accepts $\mathbf{x}_e$ as the approximate sample.

definition: Given a solution trajectory $\{\mathbf{x}_t\}_{t\in[\epsilon,T]}$ of the PF ODE in Eq. (13.26), we define the *consistency function* as $f : (\mathbf{x}_t, t) \mapsto \mathbf{x}_\epsilon$. A consistency function has the property of *self-consistency*: its outputs are consistent for arbitrary pairs of $(\mathbf{x}_t, t)$ that belong to the same PF ODE trajectory, i.e., $f(\mathbf{x}_t, t) = f(\mathbf{x}_{t'}, t')$ for all $t, t' \in [\epsilon, T]$.

The goal of a consistency model, symbolized as f_θ, is to estimate this consistency function f from data by learning to enforce the self-consistency property. Note that a similar definition is used for neural flows, and the invertibility constraint is not required here.

Parameterization For any consistency function $f(\cdot,\cdot)$, we have $f(\mathbf{x}_\epsilon, \epsilon) = \mathbf{x}_\epsilon$, i.e., $f(\cdot, \epsilon)$ is an identity function. We call this constraint the *boundary condition*. All consistency models have to meet this boundary condition, as it plays a crucial role in the successful training of consistency models.

The first way is to simply parameterize the consistency model as

$$f_\theta(\mathbf{x}, t) = \begin{cases} \mathbf{x}, & t = \epsilon \\ F_\theta(\mathbf{x}, t), & t \in (\epsilon, T] \end{cases}. \tag{13.28}$$

The second method is to parameterize the consistency model using skip connections, that is,

$$f_\theta(\mathbf{x}, t) = c_{\text{skip}}(t)\mathbf{x} + c_{\text{out}}(t) F_\theta(\mathbf{x}, t), \tag{13.29}$$

where $c_{\text{skip}}(t)$ and $c_{\text{out}}(t)$ are differentiable functions such that $c_{\text{skip}}(\epsilon) = 1$, and $c_{\text{out}}(\epsilon) = 0$. This way, the consistency model is differentiable at $t = \epsilon$ if $F_\theta(\mathbf{x}, t)$, $c_{\text{skip}}(t)$, and $c_{\text{out}}(t)$ are all differentiable, which is critical for training continuous-time consistency models. The parameterization in Eq. (13.29) bears a strong resemblance to diffusion models, making it easier to borrow powerful diffusion model architectures for constructing consistency models.

13.4.2 Training Consistency Models via Distillation

We present our first method for training consistency models based on distilling a pre-trained score model $s_\phi(\mathbf{x}, t)$. Our discussion revolves around the empirical PF ODE in Eq. (13.27), obtained by plugging the score model $s_\phi(\mathbf{x}, t)$ into the PF ODE. Consider discretizing the time horizon $[\epsilon, T]$ into $N-1$ sub-intervals, with boundaries $t_1 = \epsilon < t_2 < \cdots < t_N = T$. In practice, we determine the boundaries with the formula

$$t_i = (\epsilon^{1/\rho} + i - 1/N - 1(T^{1/\rho} - \epsilon^{1/\rho}))^\rho, \quad \rho = 7. \tag{13.30}$$

When N is sufficiently large, we can obtain an accurate estimate of $\mathbf{x}_{t_n}$ from $\mathbf{x}_{t_{n+1}}$ by running one discretization step of a numerical ODE solver. This estimate, which we denote as $\hat{\mathbf{x}}^{\phi}_{t_n}$, is defined by

$$\hat{\mathbf{x}}^{\phi}_{t_n} := \mathbf{x}_{t_{n+1}} + (t_n - t_{n+1})\Phi(\mathbf{x}_{t_n+1}, t_{n+1}; \phi), \tag{13.31}$$

where $\Phi(\cdot)$ represents the update function of a one-step ODE solver applied to the empirical PF ODE. For example, when using the Euler solver, we have $\Phi(\mathbf{x}, t; \phi) = -ts_{\phi}(\mathbf{x}, t)$, which corresponds to the following update rule:

Definition 13.1 *The consistency distillation loss is defined as*

$$\mathcal{L}^{N}_{\mathrm{CD}}(\theta, \theta^{-}; \phi) := \mathbb{E}[\mathbb{E}_{\mathbf{x}_{t_n}} d(f_{\theta}(\mathbf{x}_{t_{n+1}}, t_{n+1}), f_{\theta^{-}}(\hat{\mathbf{x}}^{\phi}_{t_n}, t_n))], \tag{13.32}$$

where the expectation is taken with respect to $x \sim p_{\mathrm{data}}$, $n \sim \mathcal{U}[1, N-1]$, and $\mathbf{x}_{t_{n+1}} \sim \mathcal{N}(\mathbf{x}; t^2_{n+1} I)$. Here, $\mathcal{U}[1, N-1]$ denotes the uniform distribution over $\{1, 2, \ldots, N-1\}$, $\omega(t_n)$ is a positive weighting function, $\hat{\mathbf{x}}^{\phi}_{t_n}$ is given by Eq. (13.31), θ^{-} denotes a running average of the past values of θ during the course of optimization, and $d(\cdot, \cdot)$ is a metric function that satisfies $d(\mathbf{x}, \mathbf{y}) \geq 0$ and $d(\mathbf{x}, \mathbf{y}) = 0$ if and only if $\mathbf{x} = \mathbf{y}$.