

Deep Generative Models

Lecture 11

Roman Isachenko

Moscow Institute of Physics and Technology
Yandex School of Data Analysis

2024, Autumn

Recap of previous lecture

$$\mathcal{L}_t = \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\tilde{\beta}_t} \left\| \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t) \right\|^2 \right]$$

Reparametrization

$$\begin{aligned}\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon} \\ \boldsymbol{\mu}_{\theta,t}(\mathbf{x}_t) &= \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \boldsymbol{\epsilon}_{\theta,t}(\mathbf{x}_t)\end{aligned}$$

$$\mathcal{L}_t = \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\frac{(1 - \alpha_t)^2}{2\tilde{\beta}_t\alpha_t(1 - \bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta,t}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}) \right\|^2 \right]$$

At each step of reverse diffusion process we try to predict the noise $\boldsymbol{\epsilon}$ that we used in the forward diffusion process!

Simplified objective

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{t \sim U\{2, T\}} \mathbb{E}_{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta,t}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}) \right\|^2$$

Recap of previous lecture

Training of DDPM

1. Get the sample $\mathbf{x}_0 \sim \pi(\mathbf{x})$.
2. Sample timestamp $t \sim U\{1, T\}$ and the noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.
3. Get noisy image $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$.
4. Compute loss $\mathcal{L}_{\text{simple}} = \|\epsilon - \epsilon_{\theta,t}(\mathbf{x}_t)\|^2$.

Sampling of DDPM

1. Sample $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Compute mean of $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \theta) = \mathcal{N}(\mu_{\theta,t}(\mathbf{x}_t), \sigma_t^2 \cdot \mathbf{I})$:

$$\mu_{\theta,t}(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \epsilon_{\theta,t}(\mathbf{x}_t)$$

3. Get denoised image $\mathbf{x}_{t-1} = \mu_{\theta,t}(\mathbf{x}_t) + \sigma_t \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.

Recap of previous lecture

DDPM objective

$$\mathbb{E}_{\pi(\mathbf{x}_0)} \mathbb{E}_{t \sim U\{1, T\}} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left[\frac{(1 - \alpha_t)^2}{2\tilde{\beta}_t \alpha_t} \left\| \mathbf{s}_{\theta, t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \right\|_2^2 \right]$$

In practice the coefficient is omitted.

NCSN objective

$$\mathbb{E}_{\pi(\mathbf{x}_0)} \mathbb{E}_{t \sim U\{1, T\}} \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \left\| \mathbf{s}_{\theta, \sigma_t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log q(\mathbf{x}_t | \mathbf{x}_0) \right\|_2^2$$

Note: The objective of DDPM and NCSN is almost identical. But the difference in sampling scheme:

- ▶ NCSN uses annealed Langevin dynamics;
- ▶ DDPM uses ancestral sampling.

Recap of previous lecture

DDPM vs NCSN: summary

- ▶ Different Markov chains:
 - ▶ DDPM: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon;$
 - ▶ NCSN: $\mathbf{x}_t = \mathbf{x}_0 + \sigma_t \cdot \epsilon.$
 - ▶ It is possible to consider the more general framework $q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\alpha_t \cdot \mathbf{x}_0, \sigma_t^2 \cdot \mathbf{I})$
- ▶ Identical objectives: ELBO \equiv score-matching.
- ▶ Different sampling schemes:
 - ▶ ancestral sampling for DDPM;
 - ▶ annealed Langevin dynamics for NCSN;
 - ▶ there is a combined approach with alternating updates of DDPM and NCSN.

Kingma D. et al. Variational Diffusion Models, 2021

Song Y. et al. Score-Based Generative Modeling through Stochastic Differential Equations, 2020

Recap of previous lecture

Unconditional generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta}) + \sigma_t \cdot \epsilon$$

Conditional generation

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}} \cdot \mathbf{x}_t + \frac{\beta_t}{\sqrt{1-\beta_t}} \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}, \boldsymbol{\theta}) + \sigma_t \cdot \epsilon$$

Conditional distribution

$$\begin{aligned} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}, \boldsymbol{\theta}) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) + \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta}) \\ &= \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) - \frac{\epsilon_{\theta,t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}} \end{aligned}$$

Here $p(\mathbf{y}|\mathbf{x}_t)$ – classifier on noisy samples (we have to learn it separately).

Recap of previous lecture

Classifier-corrected noise prediction

$$\epsilon_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \epsilon_{\theta,t}(\mathbf{x}_t) - \sqrt{1 - \bar{\alpha}_t} \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

Guidance scale

$$\epsilon_{\theta,t}(\mathbf{x}_t, \mathbf{y}) = \epsilon_{\theta,t}(\mathbf{x}_t) - \gamma \cdot \sqrt{1 - \bar{\alpha}_t} \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t)$$

- ▶ Train DDPM as usual.
- ▶ Train the additional classifier $p(\mathbf{y}|\mathbf{x}_t)$ on the noisy samples \mathbf{x}_t .

Guided sampling

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \cdot \mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{\alpha_t(1 - \bar{\alpha}_t)}} \cdot \epsilon_{\theta,t}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \epsilon$$

Note: Guidance scale γ tries to sharpen the distribution $p(\mathbf{y}|\mathbf{x}_t)$ (in this case Z should not depend on \mathbf{x}_t).

Recap of previous lecture

- ▶ Previous method requires training the additional classifier model $p(\mathbf{y}|\mathbf{x}_t)$ on the noisy data.
- ▶ Let try to avoid this requirement.

$$\nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}, \boldsymbol{\theta}) - \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta})$$

$$\begin{aligned}\nabla_{\mathbf{x}_t}^\gamma \log p(\mathbf{x}_t|\mathbf{y}, \boldsymbol{\theta}) &= \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta}) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{y}|\mathbf{x}_t) = \\ &= (1 - \gamma) \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta}) + \gamma \cdot \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}, \boldsymbol{\theta})\end{aligned}$$

Classifier-free-corrected noise prediction

$$\hat{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) = \gamma \cdot \epsilon_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) + (1 - \gamma) \cdot \epsilon_{\boldsymbol{\theta},t}(\mathbf{x}_t)$$

- ▶ Train the single model $\epsilon_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y})$ on **supervised** data alternating with real conditioning \mathbf{y} and empty conditioning $\mathbf{y} = \emptyset$.
- ▶ Apply the model twice during inference.

Outline

1. Continuous-in-time normalizing flows
2. Kolmogorov-Fokker-Planck equation for NF log-likelihood
3. FFJORD (Hutchinson's trace estimator)
4. Adjoint method for continuous-in-time NF
5. SDE basics

Outline

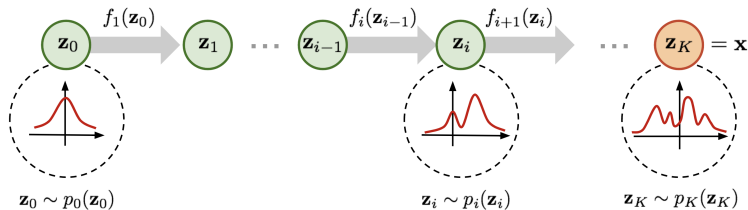
1. Continuous-in-time normalizing flows
2. Kolmogorov-Fokker-Planck equation for NF log-likelihood
3. FFJORD (Hutchinson's trace estimator)
4. Adjoint method for continuous-in-time NF
5. SDE basics

Discrete-in-time NF

Change of variable theorem (CoV)

Let \mathbf{x} be a random variable with density function $p(\mathbf{x})$ and $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^m$ is a differentiable, **invertible** function. If $\mathbf{z} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} = \mathbf{f}^{-1}(\mathbf{z}) = \mathbf{g}(\mathbf{z})$, then

$$p(\mathbf{x}) = p(\mathbf{z}) |\det(\mathbf{J}_{\mathbf{f}})| = p(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| = p(\mathbf{f}(\mathbf{x})) \left| \det \left(\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} \right) \right|$$



$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{f}_K \circ \dots \circ \mathbf{f}_1(\mathbf{x})) + \sum_{k=1}^K \log \left| \det \left(\frac{\partial \mathbf{f}_k}{\partial \mathbf{f}_{k-1}} \right) \right|.$$

Discrete-in-time NF

- ▶ Previously we assumed that the time axis is discrete:

$$\mathbf{x}_{t+1} = \mathbf{f}_\theta(\mathbf{x}_t, t); \quad \log p(\mathbf{x}_{t+1}) = \log p(\mathbf{x}_t) - \log \left| \det \frac{\partial \mathbf{f}_\theta(\mathbf{x}_t)}{\partial \mathbf{x}_t} \right|.$$

- ▶ Let consider the more general case of continuous time. It means that we will have the function $\mathbf{x}(t) : \mathbb{R} \rightarrow \mathbb{R}^m$ of continuous dynamic.

Continuous-in-time dynamics

Consider Ordinary Differential Equation (ODE)

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_\theta(\mathbf{x}(t), t) dt + \mathbf{x}_0$$

Here $\mathbf{f}_\theta : \mathbb{R}^m \times [t_0, t_1] \rightarrow \mathbb{R}^m$ is a vector field.

Numerical solution of ODE

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\theta}(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0.$$

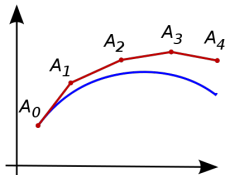
$$\mathbf{x}(t_1) = \int_{t_0}^{t_1} \mathbf{f}_{\theta}(\mathbf{x}(t), t) dt + \mathbf{x}_0 \approx \text{ODESolve}_f(\mathbf{x}_0, \theta, t_0, t_1).$$

Here we need to define the computational procedure $\text{ODESolve}_f(\mathbf{x}_0, \theta, t_0, t_1)$.

Euler update step

$$\frac{\mathbf{x}(t + \Delta t) - \mathbf{x}(t)}{\Delta t} = \mathbf{f}_{\theta}(\mathbf{x}(t), t)$$

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{f}_{\theta}(\mathbf{x}(t), t)$$



Note: Euler method is the simplest version of the ODESolve that is unstable in practice.

Continuous-in-time normalizing flows

Let consider time interval $[t_0, t_1] = [0, 1]$ without loss of generality.

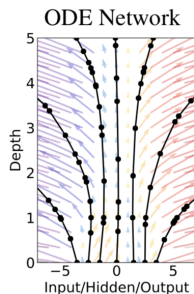
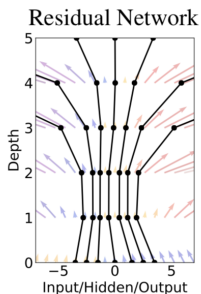
Neural ODE

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

Euler ODESolve

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \Delta t \cdot \mathbf{f}_\theta(\mathbf{x}(t), t)$$

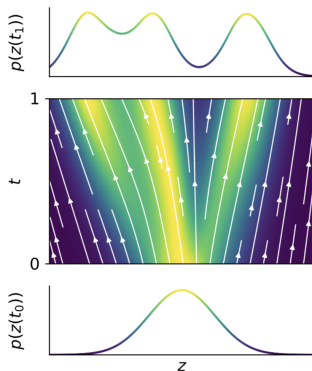
Note: It is possible to use more sophisticated numerical methods instead if Euler (e.x. Runge-Kutta methods).



Continuous-in-time Normalizing Flows

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_\theta(\mathbf{x}(t), t); \quad \text{with initial condition } \mathbf{x}(t_0) = \mathbf{x}_0$$

- ▶ $\mathbf{x}(0)$ is a random variable with the density function $p(\mathbf{x}(0))$.
- ▶ $\mathbf{x}(1)$ is a random variable with the density function $p(\mathbf{x}(1))$.
- ▶ $p_t(\mathbf{x}) = p(\mathbf{x}, t)$ is the **probability path** between $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$.
- ▶ Let say that $p_0(\mathbf{x}) = \mathcal{N}(0, \mathbf{I})$ is the base distribution and $p_1(\mathbf{x}) = \pi(\mathbf{x})$ is the data distribution (that we try to approximate with the model distribution $p(\mathbf{x}|\theta)$).



What is the difference between $p_t(\mathbf{x}(t))$ and $p_t(\mathbf{x})$?

Grathwohl W. et al. *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models*, 2018

Continuous-in-time Normalizing Flows

Theorem (Picard)

If \mathbf{f} is uniformly Lipschitz continuous in \mathbf{x} and continuous in t , then the ODE has a **unique** solution.

It means that we are able **uniquely revert** our ODE.

$$\mathbf{x} = \mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_\theta(\mathbf{x}(t), t) dt$$

$$\mathbf{x} = \mathbf{x}(0) = \mathbf{x}(1) + \int_1^0 \mathbf{f}_\theta(\mathbf{x}(t), t) dt$$

Note: Unlike discrete-in-time NF, \mathbf{f} does not need to be invertible (uniqueness guarantees bijectivity).

What is left?

- ▶ We need the way to compute $p_t(\mathbf{x})$ at any moment t .
- ▶ We need the way to find the optimal parameters θ of the dynamic \mathbf{f}_θ .

Outline

1. Continuous-in-time normalizing flows
2. Kolmogorov-Fokker-Planck equation for NF log-likelihood
3. FFJORD (Hutchinson's trace estimator)
4. Adjoint method for continuous-in-time NF
5. SDE basics

Continuous-in-time Normalizing Flows

Theorem (continuity equation)

If \mathbf{f} is uniformly Lipschitz continuous in \mathbf{x} and continuous in t , then

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right)$$

It means that if we have the value $\mathbf{x}_0 = \mathbf{x}(0)$ then the solution of the continuity equation will give us the density $p_1(\mathbf{x}(1))$.

Solution of continuity equation

$$\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0)) - \int_0^1 \text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right) dt.$$

Note: This solution will give us the density along the trajectory (not the total probability path).

Continuous-in-time Normalizing Flows

Forward transform + log-density

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_\theta(\mathbf{x}(t), t) dt$$

$$\log p_1(\mathbf{x}(1)|\theta) = \log p_0(\mathbf{x}(0)) - \int_0^1 \text{tr} \left(\frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right) dt$$

- ▶ **Discrete-in-time NF**: evaluation of determinant of the Jacobian costs $O(m^3)$ (we need invertible \mathbf{f}).
- ▶ **Continuous-in-time NF**: getting the trace of the Jacobian costs $O(m^2)$ (we need smooth \mathbf{f}).

Why $O(m^2)$?

$\text{tr} \left(\frac{\partial \mathbf{f}_\theta(\mathbf{x}(t))}{\partial \mathbf{x}(t)} \right)$ costs $O(m^2)$ (m evaluations of \mathbf{f}), since we have to compute a derivative for each diagonal element. It is possible to reduce cost from $O(m^2)$ to $O(m)$!

Outline

1. Continuous-in-time normalizing flows
2. Kolmogorov-Fokker-Planck equation for NF log-likelihood
3. FFJORD (Hutchinson's trace estimator)
4. Adjoint method for continuous-in-time NF
5. SDE basics

Continuous-in-time Normalizing Flows

Hutchinson's trace estimator

If $\epsilon \in \mathbb{R}^m$ is a random variable with $\mathbb{E}[\epsilon] = 0$ and $\text{Cov}(\epsilon) = \mathbf{I}$, then

$$\begin{aligned}\text{tr}(\mathbf{A}) &= \text{tr}(\mathbf{A} \cdot \mathbf{I}) = \text{tr}\left(\mathbf{A} \cdot \mathbb{E}_{p(\epsilon)}\left[\epsilon\epsilon^T\right]\right) = \\ &= \mathbb{E}_{p(\epsilon)}\left[\text{tr}\left(\mathbf{A}\epsilon\epsilon^T\right)\right] = \mathbb{E}_{p(\epsilon)}\left[\epsilon^T \mathbf{A} \epsilon\right]\end{aligned}$$

Jacobian vector products $\mathbf{v}^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ can be computed for approximately the same cost as evaluating \mathbf{f} (`torch.autograd.functional.jvp`).

FFJORD density estimation

$$\begin{aligned}\log p_1(\mathbf{x}(1)) &= \log p_0(\mathbf{x}(0)) - \int_0^1 \text{tr}\left(\frac{\partial \mathbf{f}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)}\right) dt = \\ &= \log p_0(\mathbf{x}(0)) - \mathbb{E}_{p(\epsilon)} \int_0^1 \left[\epsilon^T \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \epsilon\right] dt.\end{aligned}$$

Outline

1. Continuous-in-time normalizing flows
2. Kolmogorov-Fokker-Planck equation for NF log-likelihood
3. FFJORD (Hutchinson's trace estimator)
4. Adjoint method for continuous-in-time NF
5. SDE basics

Continuous-in-time NF

Dynamics ODE

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}_{\theta}(\mathbf{x}(t), t)$$

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_{\theta}(\mathbf{x}(t), t) dt$$

Continuity ODE

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr} \left(\frac{\partial \mathbf{f}_{\theta}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right)$$

$$\log p_1(\mathbf{x}(1)) = \log p_0(\mathbf{x}(0)) - \int_0^1 \text{tr} \left(\frac{\partial \mathbf{f}_{\theta}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right) dt$$

How to get optimal parameters of θ ?

- ▶ We need the gradients for fitting the parameters
- ▶ We need the continuous analogue of the backpropagation.

Neural ODE

Forward pass (Loss function)

$$L(\mathbf{x}) = -\log p_1(\mathbf{x}(1)|\theta) = -\log p_0(\mathbf{x}(0)) + \int_0^1 \text{tr} \left(\frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} \right) dt$$

Adjoint functions

$$\mathbf{a}_\mathbf{x}(t) = \frac{\partial L}{\partial \mathbf{x}(t)}; \quad \mathbf{a}_\theta(t) = \frac{\partial L}{\partial \theta(t)}.$$

These functions show how the gradient of the loss depends on the hidden state $\mathbf{x}(t)$ and parameters θ .

Theorem (Pontryagin)

$$\frac{d\mathbf{a}_\mathbf{x}(t)}{dt} = -\mathbf{a}_\mathbf{x}(t)^T \cdot \frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \mathbf{x}}; \quad \frac{d\mathbf{a}_\theta(t)}{dt} = -\mathbf{a}_\mathbf{x}(t)^T \cdot \frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \theta}.$$

Adjoint method

Theorem (Pontryagin)

$$\frac{d\mathbf{a}_x(t)}{dt} = -\mathbf{a}_x(t)^T \cdot \frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \mathbf{x}}; \quad \frac{d\mathbf{a}_\theta(t)}{dt} = -\mathbf{a}_x(t)^T \cdot \frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \theta}.$$

Solution for the adjoints function

$$\begin{aligned} \frac{\partial L}{\partial \theta(0)} = \mathbf{a}_\theta(0) &= - \int_1^0 \mathbf{a}_x(t)^T \frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \theta(t)} dt + 0 \\ \frac{\partial L}{\partial \mathbf{x}(0)} = \mathbf{a}_x(0) &= - \int_1^0 \mathbf{a}_x(t)^T \frac{\partial \mathbf{f}_\theta(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} dt + \frac{\partial L}{\partial \mathbf{x}(1)} \end{aligned}$$

- ▶ Think about the initial conditions.
- ▶ These equations are solved in the reverse time direction.
- ▶ Numerical solvers (Euler ODEsolve) are used to solve them.

Adjoint method

Forward pass

$$\mathbf{x}(1) = \mathbf{x}(0) + \int_0^1 \mathbf{f}_{\theta}(\mathbf{x}(t), t) dt \Rightarrow \text{ODE Solver}$$

Backward pass

$$\left. \begin{aligned} \frac{\partial L}{\partial \theta(0)} = \mathbf{a}_{\theta}(0) &= - \int_1^0 \mathbf{a}_{\mathbf{x}}(t)^T \frac{\partial \mathbf{f}_{\theta}(\mathbf{x}(t), t)}{\partial \theta(t)} dt + 0 \\ \frac{\partial L}{\partial \mathbf{x}(0)} = \mathbf{a}_{\mathbf{x}}(0) &= - \int_1^0 \mathbf{a}_{\mathbf{x}}(t)^T \frac{\partial \mathbf{f}_{\theta}(\mathbf{x}(t), t)}{\partial \mathbf{x}(t)} dt + \frac{\partial L}{\partial \mathbf{x}(1)} \\ \mathbf{x}(0) &= - \int_0^1 \mathbf{f}_{\theta}(\mathbf{x}(t), t) dt + \mathbf{x}(1). \end{aligned} \right\} \Rightarrow \text{ODE Solver}$$

Note: These scary formulas are the standard backprop in the discrete case.

Outline

1. Continuous-in-time normalizing flows
2. Kolmogorov-Fokker-Planck equation for NF log-likelihood
3. FFJORD (Hutchinson's trace estimator)
4. Adjoint method for continuous-in-time NF
5. SDE basics

Stochastic differential equation (SDE)

Let define stochastic process $\mathbf{x}(t)$ with initial condition $\mathbf{x}(0) \sim p_0(\mathbf{x}) = \pi(\mathbf{x})$:

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$$

- ▶ $\mathbf{f}(\mathbf{x}, t) : \mathbb{R}^m \times [0, 1] \rightarrow \mathbb{R}^m$ is the **drift** function of $\mathbf{x}(t)$.
- ▶ $g(t) : \mathbb{R} \rightarrow \mathbb{R}$ is the **diffusion** function of $\mathbf{x}(t)$.
- ▶ $\mathbf{w}(t)$ is the standard Wiener process (Brownian motion):
 1. $\mathbf{w}(0) = 0$ (almost surely);
 2. $\mathbf{w}(t)$ has independent increments;
 3. $\mathbf{w}(t) - \mathbf{w}(s) \sim \mathcal{N}(0, (t-s)\mathbf{I})$, for $t > s$.
- ▶ $d\mathbf{w} = \mathbf{w}(t+dt) - \mathbf{w}(t) = \mathcal{N}(0, \mathbf{I} \cdot dt) = \epsilon \cdot \sqrt{dt}$, where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$.
- ▶ If $g(t) = 0$ we get standard ODE.

Stochastic differential equation (SDE)

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$$

- ▶ In contrast to ODE, initial condition $\mathbf{x}(0)$ does not uniquely determine the process trajectory.
- ▶ We have two sources of randomness: initial distribution $p_0(\mathbf{x})$ and Wiener process $\mathbf{w}(t)$.

Discretization of SDE (Euler method)

$$\mathbf{x}(t + dt) = \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t) \cdot dt + g(t) \cdot \epsilon \cdot \sqrt{dt}$$

If $dt = 1$, then

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{f}(\mathbf{x}_t, t) + g(t) \cdot \epsilon$$

- ▶ At each moment t we have the density $p_t(\mathbf{x}) = p(\mathbf{x}, t)$.
- ▶ $p : \mathbb{R}^m \times [0, 1] \rightarrow \mathbb{R}_+$ is a **probability path** between $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$.
- ▶ How to get the distribution path $p_t(\mathbf{x})$ for $\mathbf{x}(t)$?

Stochastic differential equation (SDE)

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}, \quad d\mathbf{w} = \boldsymbol{\epsilon} \cdot \sqrt{dt}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$

Theorem (Kolmogorov-Fokker-Planck)

Evolution of the distribution $p_t(\mathbf{x})$ is given by the following equation:

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = -\operatorname{div}(\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})) + \frac{1}{2}g^2(t)\Delta_{\mathbf{x}}p_t(\mathbf{x})$$

Here

$$\operatorname{div}(\mathbf{v}) = \sum_{i=1}^m \frac{\partial v_i(\mathbf{x})}{\partial x_i} = \operatorname{tr} \left(\frac{\partial \mathbf{v}(\mathbf{x})}{\partial \mathbf{x}} \right)$$

$$\Delta_{\mathbf{x}}p_t(\mathbf{x}) = \sum_{i=1}^m \frac{\partial^2 p_t(\mathbf{x})}{\partial x_i^2} = \operatorname{tr} \left(\frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right)$$

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \operatorname{tr} \left(-\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t)p_t(\mathbf{x})] + \frac{1}{2}g^2(t) \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right)$$

Stochastic differential equation (SDE)

Theorem (Kolmogorov-Fokker-Planck)

$$\frac{\partial p_t(\mathbf{x})}{\partial t} = \text{tr} \left(-\frac{\partial}{\partial \mathbf{x}} [\mathbf{f}(\mathbf{x}, t) p_t(\mathbf{x})] + \frac{1}{2} g^2(t) \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right)$$

- ▶ KFP theorem does not define the SDE uniquely in general case.
- ▶ This is the generalization of KFP theorem that we used in continuous-in-time NF:

$$\frac{d \log p_t(\mathbf{x}(t))}{dt} = -\text{tr} \left(\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial \mathbf{x}} \right).$$

Langevin SDE (special case)

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + \mathbf{g}(t)d\mathbf{w}$$

$$d\mathbf{x} = \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) dt + \mathbf{1} \cdot d\mathbf{w}$$

Let apply KFP theorem to this SDE.

Langevin SDE (special case)

$$d\mathbf{x} = \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) dt + 1 \cdot d\mathbf{w}$$

$$\begin{aligned} \frac{\partial p_t(\mathbf{x})}{\partial t} &= \text{tr} \left(-\frac{\partial}{\partial \mathbf{x}} \left[p_t(\mathbf{x}) \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) \right] + \frac{1}{2} \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right) = \\ &= \text{tr} \left(-\frac{\partial}{\partial \mathbf{x}} \left[\frac{1}{2} \frac{\partial}{\partial \mathbf{x}} p_t(\mathbf{x}) \right] + \frac{1}{2} \frac{\partial^2 p_t(\mathbf{x})}{\partial \mathbf{x}^2} \right) = 0 \end{aligned}$$

The density $p_t(\mathbf{x}) = \text{const}(t)$!

If $\mathbf{x}(0) \sim p_0(\mathbf{x})$, then $\mathbf{x}(t) \sim p_0(\mathbf{x})$.

Discretized Langevin SDE

$$\mathbf{x}_{t+1} - \mathbf{x}_t = \frac{\eta}{2} \cdot \frac{\partial}{\partial \mathbf{x}} \log p_t(\mathbf{x}) + \sqrt{\eta} \cdot \epsilon, \quad \eta \approx dt.$$

Langevin dynamic

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \frac{\eta}{2} \cdot \nabla_{\mathbf{x}} \log p(\mathbf{x}|\theta) + \sqrt{\eta} \cdot \epsilon, \quad \eta \approx dt.$$

Summary

- ▶ Continuous-in-time NF uses neural ODE to define continuous dynamic $\mathbf{x}(t)$. It has less functional restrictions.
- ▶ Continuity equation allows to calculate $\log p(\mathbf{x}, t)$ at arbitrary moment t .
- ▶ FFJORD model makes such kind of NF scalable.
- ▶ SDE defines stochastic process with drift and diffusion terms. ODEs are the special case of SDEs.
- ▶ KFP equation defines the dynamic of the probability function for the SDE.
- ▶ Langevin SDE has constant probability path.