Deep Generative Models

Lecture 13

Roman Isachenko

AI Masters

2024, Summer

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 $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.
3. Get noisy image $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \cdot \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \boldsymbol{\epsilon}$.
4. Compute loss $\mathcal{L}_{\text{simple}} = \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)\|^2$.

Sampling of DDPM

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

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

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

Ho J. Denoising Diffusion Probabilistic Models, 2020

Recap of previous lecture

DDPM objective

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

In practice the coefficient is omitted.

NCSN objective

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

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.

$$\mathbf{s}_{\boldsymbol{\theta},t}(\mathbf{x}_t) = -\frac{\boldsymbol{\epsilon}_{\boldsymbol{\theta},t}(\mathbf{x}_t)}{\sqrt{1-\bar{\alpha}_t}} = \nabla_{\mathbf{x}_t}\log p(\mathbf{x}_t|\boldsymbol{\theta})$$

Ho J. Denoising Diffusion Probabilistic Models, 2020

Recap of previous lecture

Unconditional generation

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

Conditional generation

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

Conditional distribution

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

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

Classifier-corrected noise prediction

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

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

Recap of previous lecture

Guidance scale

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

$$\nabla^{\gamma}_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{y}, \boldsymbol{\theta}) = \nabla_{\mathbf{x}_t} \log \left(\frac{p(\mathbf{y}|\mathbf{x}_t)^{\gamma} p(\mathbf{x}_t|\boldsymbol{\theta})}{Z} \right)$$

Note: Guidance scale γ tries to sharpen the distribution $p(\mathbf{y}|\mathbf{x}_t)$.

Guided sampling

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

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

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_{\boldsymbol{\theta},t}(\mathbf{x}_t, \mathbf{y}) + \sigma_t \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

Dhariwal P., Nichol A. Diffusion Models Beat GANs on Image Synthesis, 2021

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.

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

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

Classifier-free-corrected noise prediction

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

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

Ho J., Salimans T. Classifier-Free Diffusion Guidance, 2022

Recap of previous lecture

Continuous-in-time dynamic (neural ODE)

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

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

Euler update step

$$\frac{\mathbf{z}(t + \Delta t) - \mathbf{z}(t)}{\Delta t} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) \ \Rightarrow \ \mathbf{z}(t+\Delta t) = \mathbf{z}(t) + \Delta t \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)$$

Theorem (Picard)

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

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

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

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Recap of previous lecture

Theorem (Kolmogorov-Fokker-Planck: special case)

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

$$\frac{d \log p(\mathbf{z}(t), t)}{dt} = -\mathrm{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right).$$

$$\log p(\mathbf{z}(t_1), t_1) = \log p(\mathbf{z}(t_0), t_0) - \int_{t_0}^{t_1} \mathrm{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right) dt.$$

- **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 $\mathbf{f}$).

Hutchinson's trace estimator

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\boldsymbol{\epsilon})} \int_{t_0}^{t_1} \left[\boldsymbol{\epsilon}^T \frac{\partial \mathbf{f}}{\partial \mathbf{z}} \boldsymbol{\epsilon}\right] dt.$$

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

Recap of previous lecture

Forward pass (Loss function)

$$\mathbf{z} = \mathbf{x} + \int_{t_1}^{t_0} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt, \quad L(\mathbf{z}) = -\log p(\mathbf{x}|\boldsymbol{\theta})$$

$$L(\mathbf{z}) = -\log p(\mathbf{z}) + \int_{t_0}^{t_1} \operatorname{tr} \left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)} \right) dt$$

Adjoint functions

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

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

Theorem (Pontryagin)

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

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Recap of previous lecture

Forward pass

$$\mathbf{z} = \mathbf{z}(t_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt + \mathbf{x} \quad \Rightarrow \quad \text{ODE Solver}$$

Backward pass

$$\left.\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\theta}(t_1)} &= \mathbf{a}_{\boldsymbol{\theta}}(t_1) = -\int_{t_0}^{t_1} \mathbf{a}_{\mathbf{z}}(t)^T \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \boldsymbol{\theta}(t)} dt + 0 \\
\frac{\partial L}{\partial \mathbf{z}(t_1)} &= \mathbf{a}_{\mathbf{z}}(t_1) = -\int_{t_0}^{t_1} \mathbf{a}_{\mathbf{z}}(t)^T \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)} dt + \frac{\partial L}{\partial \mathbf{z}(t_0)} \\
\mathbf{z}(t_1) &= -\int_{t_1}^{t_0} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt + \mathbf{z}_0.
\end{aligned}\right\} \Rightarrow \text{ODE Solver}$$

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

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Outline

1. Continuous-in-time normalizing flows
2. Continuous-in-time NF: adjoint method
3. SDE basics
4. Probability flow ODE
5. Reverse SDE
6. Diffusion and Score matching SDEs

Outline

Continuous-in-time normalizing flows

Discrete-in-time NF

Previously we assume that the time axis is discrete:

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

Let assume the more general case of continuous time. It means that we will have the dynamic function $\mathbf{z}(t)$.

Continuous-in-time dynamics

Consider Ordinary Differential Equation (ODE)

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

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

Here we need to define the computational procedure $\text{ODESolve}(\mathbf{z}(t_0), \mathbf{f}_{\boldsymbol{\theta}}, t_0, t_1)$.

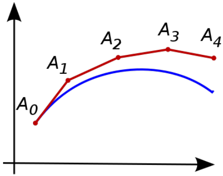
Grathwohl W. et al. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models, 2018

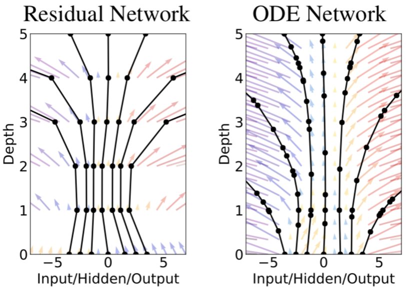
Continuous-in-time normalizing flows

Euler update step

$$\frac{\mathbf{z}(t + \Delta t) - \mathbf{z}(t)}{\Delta t} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) \;\Rightarrow\; \mathbf{z}(t+\Delta t) = \mathbf{z}(t) + \Delta t \cdot \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)$$

Note: Euler method is the simplest version of ODESolve that is unstable in practice. It is possible to use more sophisticated methods (e.x. Runge-Kutta methods).





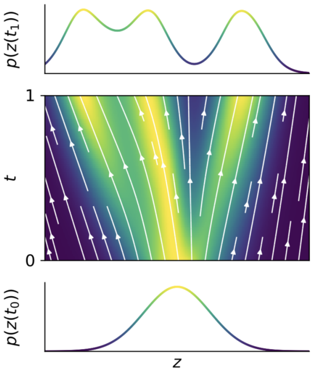
Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-in-time Normalizing Flows

Neural ODE

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

- Let $\mathbf{z}(t_0)$ will be a random variable with some density function $p(\mathbf{z}(t_0))$.
- Then $\mathbf{z}(t_1)$ will be also a random variable with some other density function $p(\mathbf{z}(t_1))$.
- We could say that we have the joint density function $p(\mathbf{z}(t), t)$.
- What is the difference between $p(\mathbf{z}(t), t)$ and $p(\mathbf{z}, t)$?



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

Continuous-in-time Normalizing Flows

Let say that $p(\mathbf{z}, t_0)$ is the base distribution (e.x. standard Normal) and $p(\mathbf{z}, t_1)$ is the desired model distribution $p(\mathbf{x}|\boldsymbol{\theta})$.

Theorem (Picard)

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

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

Forward and inverse transforms

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

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

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

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-in-time Normalizing Flows

What do we need?

- We need the way to compute $p(\mathbf{z}, t)$ at any moment t.
- We need the way to find the optimal parameters $\boldsymbol{\theta}$ of the dynamic $\mathbf{f}_{\boldsymbol{\theta}}$.

Theorem (Kolmogorov-Fokker-Planck: special case)

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

$$\frac{d \log p(\mathbf{z}(t), t)}{dt} = -\operatorname{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right).$$

$$\log p(\mathbf{z}(t_1), t_1) = \log p(\mathbf{z}(t_0), t_0) - \int_{t_0}^{t_1} \operatorname{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right) dt.$$

It means that if we have the value $\mathbf{z}_0 = \mathbf{z}(t_0)$ then the solution of the ODE will give us the density at the moment t_1.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-in-time Normalizing Flows

Forward transform + log-density

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

$$\log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{z}) - \int_{t_0}^{t_1} \mathrm{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right) dt$$

Here $p(\mathbf{x}|\boldsymbol{\theta}) = p(\mathbf{z}(t_1), t_1)$, $p(\mathbf{z}) = p(\mathbf{z}(t_0), t_0)$.

- **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 $O(m^2)$?

$\mathrm{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t))}{\partial \mathbf{z}(t)}\right)$ 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 $O(m)$!

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Continuous-in-time Normalizing Flows

Hutchinson's trace estimator

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

$$\mathrm{tr}(\mathbf{A}) = \mathrm{tr}\left(\mathbf{A} \cdot \mathbf{I}\right) = \mathrm{tr}\left(\mathbf{A} \cdot \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right]\right) =$$

$$= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\mathrm{tr}\left(\mathbf{A}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right)\right] = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\epsilon}\right]$$

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

FFJORD density estimation

$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right) dt =$$

$$= \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\boldsymbol{\epsilon})} \int_{t_0}^{t_1} \left[\boldsymbol{\epsilon}^T \frac{\partial \mathbf{f}}{\partial \mathbf{z}} \boldsymbol{\epsilon}\right] dt.$$

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

Outline

1. Continuous-in-time normalizing flows
2. Continuous-in-time NF: adjoint method
3. SDE basics
4. Probability flow ODE
5. Reverse SDE
6. Diffusion and Score matching SDEs

Neural ODE

Continuous-in-time NF

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) \qquad \frac{d \log p(\mathbf{z}(t), t)}{dt} = -\text{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right)$$

$$\mathbf{x} = \mathbf{z} + \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt \quad \log p(\mathbf{x}|\boldsymbol{\theta}) = \log p(\mathbf{z}) - \int_{t_0}^{t_1} \text{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right) dt$$

How to get optimal parameters of $\boldsymbol{\theta}$?
For fitting parameters we need gradients. We need the analogue of the backpropagation.

Forward pass (Loss function)

$$\mathbf{z} = \mathbf{x} + \int_{t_1}^{t_0} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt, \quad L(\mathbf{z}) = -\log p(\mathbf{x}|\boldsymbol{\theta})$$

$$L(\mathbf{z}) = -\log p(\mathbf{z}) + \int_{t_0}^{t_1} \text{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right) dt$$

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Neural ODE

Adjoint functions

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

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

Theorem (Pontryagin)

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

Solution for adjoint function

$$\frac{\partial L}{\partial \boldsymbol{\theta}(t_1)} = \mathbf{a}_{\boldsymbol{\theta}}(t_1) = -\int_{t_0}^{t_1} \mathbf{a_z}(t)^T \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \boldsymbol{\theta}(t)} dt + 0$$

$$\frac{\partial L}{\partial \mathbf{z}(t_1)} = \mathbf{a_z}(t_1) = -\int_{t_0}^{t_1} \mathbf{a_z}(t)^T \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)} dt + \frac{\partial L}{\partial \mathbf{z}(t_0)}$$

Note: These equations are solved in reverse time direction.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Adjoint method

Forward pass

$$\mathbf{z} = \mathbf{z}(t_0) = \int_{t_0}^{t_1} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt + \mathbf{x} \quad \Rightarrow \quad \text{ODE Solver}$$

Backward pass

$$\left.\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\theta}(t_1)} &= \mathbf{a}_{\boldsymbol{\theta}}(t_1) = -\int_{t_0}^{t_1} \mathbf{a}_{\mathbf{z}}(t)^T \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \boldsymbol{\theta}(t)} dt + 0 \\
\frac{\partial L}{\partial \mathbf{z}(t_1)} &= \mathbf{a}_{\mathbf{z}}(t_1) = -\int_{t_0}^{t_1} \mathbf{a}_{\mathbf{z}}(t)^T \frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)} dt + \frac{\partial L}{\partial \mathbf{z}(t_0)} \\
\mathbf{z}(t_1) &= -\int_{t_1}^{t_0} \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) dt + \mathbf{z}_0.
\end{aligned}\right\} \Rightarrow \text{ODE Solver}$$

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

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

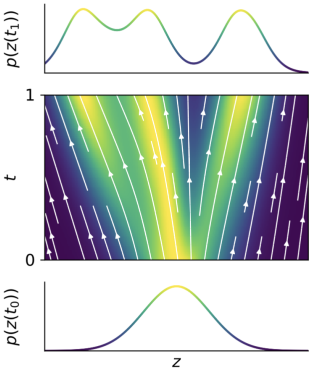
Outline

Ordinary differential equation (ODE)

Continuous-in-time Normalizing Flows

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

- Let $\mathbf{z}(t_0)$ will be a random variable with some density function $p(\mathbf{z}(t_0))$.
- Then $\mathbf{z}(t_1)$ will be also a random variable with some other density function $p(\mathbf{z}(t_1))$.
- We could say that we have the joint density function $p(\mathbf{z}(t), t)$.
- What is the difference between $p(\mathbf{z}(t), t)$ and $p(\mathbf{z}, t)$?



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

Ordinary differential equation (ODE)

$$d\mathbf{z} = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}, t) \cdot dt$$

Discretization of ODE (Euler method)

$$\mathbf{z}(t + dt) = \mathbf{z}(t) + \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t) \cdot dt$$

Theorem (Kolmogorov-Fokker-Planck: special case)

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

$$\frac{d \log p(\mathbf{z}(t), t)}{dt} = -\mathrm{tr}\left(\frac{\partial \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{z}(t), t)}{\partial \mathbf{z}(t)}\right).$$

It means that if we have the value $\mathbf{z}_0 = \mathbf{z}(t_0)$ then the solution of the ODE will give us the density at the moment t_1.

Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

Stochastic differential equation (SDE)

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

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

- $\mathbf{f}(\mathbf{x}, t) : \mathbb{R}^m \times [0, 1] \to \mathbb{R}^m$ is the **drift** function of $\mathbf{x}(t)$.
- $g(t) : \mathbb{R} \to \mathbb{R}$ 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 $t > s$.
- $d\mathbf{w} = \mathbf{w}(t + dt) - \mathbf{w}(t) = \mathcal{N}(0, \mathbf{I} \cdot dt) = \boldsymbol{\epsilon} \cdot \sqrt{dt}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.
- 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 $p(\mathbf{x}(t), t)$.
- $p : \mathbb{R}^m \times [0, 1] \to \mathbb{R}_+$ is a **probability path** between $p_0(\mathbf{x})$ and $p_1(\mathbf{x})$.
- How to get the distribution path $p(\mathbf{x}, t)$ 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 $p(\mathbf{x}, t)$ is given by the following equation:

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = -\mathrm{div}\left(\mathbf{f}(\mathbf{x}, t)p(\mathbf{x}, t)\right) + \frac{1}{2}g^2(t)\Delta_{\mathbf{x}}p(\mathbf{x}, t)$$

Here

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

$$\Delta_{\mathbf{x}}p(\mathbf{x}, t) = \sum_{i=1}^{m} \frac{\partial^2 p(\mathbf{x}, t)}{\partial x_i^2} = \mathrm{tr}\left(\frac{\partial^2 p(\mathbf{x}, t)}{\partial \mathbf{x}^2}\right)$$

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

Stochastic differential equation (SDE)

Theorem (Kolmogorov-Fokker-Planck)

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

- KFP theorem uniquely defines the SDE.
- This is the generalization of KFP theorem that we used in continuous-in-time NF:

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

Langevin SDE (special case)

$$d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$$
$$d\mathbf{x} = \frac{1}{2}\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, t)dt + 1 \cdot d\mathbf{w}$$

Let apply KFP theorem to this SDE.

Langevin SDE (special case)

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

$$\frac{\partial p(\mathbf{x}, t)}{\partial t} = \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\left[p(\mathbf{x}, t)\frac{1}{2}\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)\right] + \frac{1}{2}\frac{\partial^2 p(\mathbf{x}, t)}{\partial \mathbf{x}^2}\right) =$$

$$= \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\left[\frac{1}{2}\frac{\partial}{\partial \mathbf{x}} p(\mathbf{x}, t)\right] + \frac{1}{2}\frac{\partial^2 p(\mathbf{x}, t)}{\partial \mathbf{x}^2}\right) = 0$$

The density $p(\mathbf{x}, t) = \mathrm{const}(t)$!
If $\mathbf{x}(0) \sim p_0(\mathbf{x})$, then $\mathbf{x}(t) \sim p_0(\mathbf{x})$.

Discretized Langevin SDE

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

Langevin dynamic

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

Outline

1. Continuous-in-time normalizing flows
2. Continuous-in-time NF: adjoint method
3. SDE basics
4. Probability flow ODE
5. Reverse SDE
6. Diffusion and Score matching SDEs

Probability flow ODE

Theorem

Assume SDE $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ induces the probability path $p(\mathbf{x}, t)$. Then there exists ODE with identical probability path $p(\mathbf{x}, t)$ of the form

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)\right] dt$$

Proof

$$\begin{aligned}
\frac{\partial p(\mathbf{x}, t)}{\partial t} &= \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\big[\mathbf{f}(\mathbf{x}, t)p(\mathbf{x}, t)\big] + \frac{1}{2}g^2(t)\frac{\partial^2 p(\mathbf{x}, t)}{\partial \mathbf{x}^2}\right) = \\
&= \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\left[\mathbf{f}(\mathbf{x}, t)p(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial p(\mathbf{x}, t)}{\partial \mathbf{x}}\right]\right) = \\
&= \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\left[\mathbf{f}(\mathbf{x}, t)p(\mathbf{x}, t) - \frac{1}{2}g^2(t)p(\mathbf{x}, t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right]\right) = \\
&= \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\left[\left(\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right) p(\mathbf{x}, t)\right]\right)
\end{aligned}$$

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

Probability flow ODE

Theorem

Assume SDE $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ induces the distribution $p(\mathbf{x}, t)$. Then there exists ODE with identical probabilities distribution $p(\mathbf{x}, t)$ of the form

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)\right] dt$$

Proof (continued)

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

$$= \mathrm{tr}\left(-\frac{\partial}{\partial \mathbf{x}}\left[\tilde{\mathbf{f}}(\mathbf{x}, t)p(\mathbf{x}, t)\right]\right)$$

$$d\mathbf{x} = \tilde{\mathbf{f}}(\mathbf{x}, t)dt + 0 \cdot d\mathbf{w} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)\right] dt$$

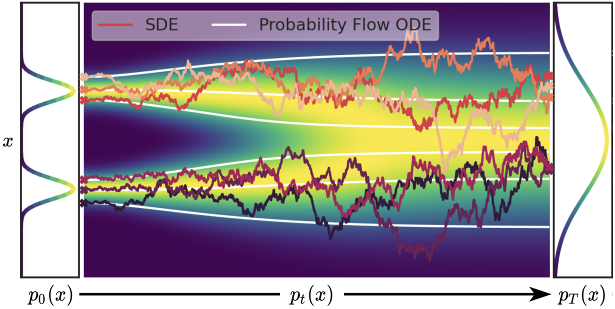
Song Y., et al. Score-Based Generative Modeling through Stochastic Differential Equations, 2020

Probability flow ODE

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

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)\right] dt \text{ -- probability flow ODE}$$

- The term $\mathbf{s}(\mathbf{x}, t) = \frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)$ is a score function for continuous time.
- ODE has more stable trajectories.



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

Outline

Reverse SDE

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

Here dt could be > 0 or < 0.

Reverse ODE

Let $\tau = 1 - t$ ($d\tau = -dt$).

$$d\mathbf{x} = -\mathbf{f}(\mathbf{x}, 1 - \tau)d\tau$$

- How to revert SDE $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$?
- Wiener process gives the randomness that we have to revert.

Theorem

There exists the reverse SDE for the SDE $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ that has the following form

$$d\mathbf{x} = \left(\mathbf{f}(\mathbf{x}, t) - g^2(t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right) dt + g(t)d\mathbf{w}$$

with $dt < 0$.

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

Reverse SDE

Theorem

There exists the reverse SDE for the SDE $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ that has the following form

$$d\mathbf{x} = \left(\mathbf{f}(\mathbf{x}, t) - g^2(t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right) dt + g(t)d\mathbf{w}$$

with $dt < 0$.

Note: Here we also see the score function $\mathbf{s}(\mathbf{x}, t) = \frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, t)$.

Sketch of the proof

- Convert initial SDE to probability flow ODE.
- Revert probability flow ODE.
- Convert reverse probability flow ODE to reverse SDE.

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

Reverse SDE

Proof

- Convert initial SDE to probability flow ODE

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

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, t)\right]dt$$

- Revert probability flow ODE

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, t)\right]dt$$

$$d\mathbf{x} = \left[-\mathbf{f}(\mathbf{x}, 1-\tau) + \frac{1}{2}g^2(1-\tau)\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, 1-\tau)\right]d\tau$$

- Convert reverse probability flow ODE to reverse SDE

$$d\mathbf{x} = \left[-\mathbf{f}(\mathbf{x}, 1-\tau) + \frac{1}{2}g^2(1-\tau)\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, 1-\tau)\right]d\tau$$

$$d\mathbf{x} = \left[-\mathbf{f}(\mathbf{x}, 1-\tau) + g^2(1-\tau)\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, 1-\tau)\right]d\tau + g(1-\tau)d\mathbf{w}$$

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

Reverse SDE

Theorem

There exists the reverse SDE for the SDE $d\mathbf{x} = \mathbf{f}(\mathbf{x}, t)dt + g(t)d\mathbf{w}$ that has the following form

$$d\mathbf{x} = \left(\mathbf{f}(\mathbf{x}, t) - g^2(t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right) dt + g(t)d\mathbf{w}$$

with $dt < 0$.

Proof (continued)

$$d\mathbf{x} = \left[-\mathbf{f}(\mathbf{x}, 1 - \tau) + g^2(1 - \tau)\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}, 1 - \tau)\right] d\tau + g(1 - \tau)d\mathbf{w}$$

$$d\mathbf{x} = \left(\mathbf{f}(\mathbf{x}, t) - g^2(t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right) dt + g(t)d\mathbf{w}$$

Here $d\tau > 0$ and $dt < 0$.

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

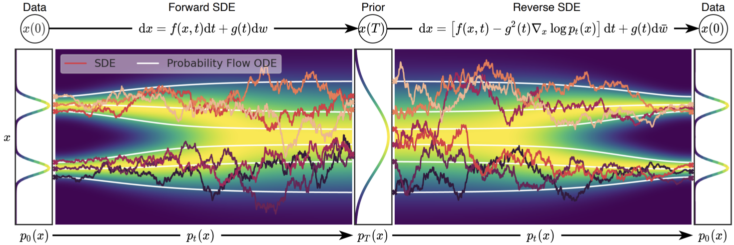
Reverse SDE

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

$$d\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - \frac{1}{2}g^2(t)\frac{\partial}{\partial \mathbf{x}}\log p(\mathbf{x}, t)\right] dt \text{ -- probability flow ODE}$$

$$d\mathbf{x} = \left(\mathbf{f}(\mathbf{x}, t) - g^2(t)\frac{\partial \log p(\mathbf{x}, t)}{\partial \mathbf{x}}\right) dt + g(t)d\mathbf{w} \text{ -- reverse SDE}$$

- We got the way to transform one distribution to another via SDE with some probability path $p(\mathbf{x}, t)$.
- We are able to revert this process with the score function.



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

Outline

Score matching SDE

Denoising score matching

$$\mathbf{x}_t = \mathbf{x} + \sigma_t \cdot \boldsymbol{\epsilon}_t, \quad p(\mathbf{x}, \sigma_t) = \mathcal{N}(\mathbf{x}, \sigma_t^2 \cdot \mathbf{I})$$
$$\mathbf{x}_{t-1} = \mathbf{x} + \sigma_{t-1} \cdot \boldsymbol{\epsilon}_{t-1}, \quad p(\mathbf{x}, \sigma_{t-1}) = \mathcal{N}(\mathbf{x}, \sigma_{t-1}^2 \cdot \mathbf{I})$$

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2} \cdot \boldsymbol{\epsilon}, \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_{t-1}, (\sigma_t^2 - \sigma_{t-1}^2) \cdot \mathbf{I})$$

Let turn this Markov chain to the continuous stochastic process $\mathbf{x}(t)$ taking $T \to \infty$:

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

Variance Exploding SDE

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

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

Diffusion SDE

Denoising Diffusion

$$\mathbf{x}_t = \sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1} + \sqrt{\beta_t}\cdot\boldsymbol{\epsilon}, \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\cdot\mathbf{x}_{t-1}, \beta_t\cdot\mathbf{I})$$

Let turn this Markov chain to the continuous stochastic process taking $T \to \infty$ and taking $\beta(\frac{t}{T}) = \beta_t \cdot T$

$$\begin{aligned}\mathbf{x}(t) &= \sqrt{1-\beta(t)dt}\cdot\mathbf{x}(t-dt) + \sqrt{\beta(t)dt}\cdot\boldsymbol{\epsilon} \approx \\ &\approx (1-\frac{1}{2}\beta(t)dt)\cdot\mathbf{x}(t-dt) + \sqrt{\beta(t)dt}\cdot\boldsymbol{\epsilon} = \\ &= \mathbf{x}(t-dt) - \frac{1}{2}\beta(t)\mathbf{x}(t-dt)dt + \sqrt{\beta(t)}\cdot d\mathbf{w}\end{aligned}$$

Variance Preserving SDE

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

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

Diffusion SDE

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

Variance Exploding SDE (NCSN)

$$d\mathbf{x} = \sqrt{\frac{d[\sigma^2(t)]}{dt}} \cdot d\mathbf{w}, \quad \mathbf{f}(\mathbf{x}, t) = 0, \quad g(t) = \sqrt{\frac{d[\sigma^2(t)]}{dt}}$$

Variance grows since $\sigma(t)$ is a monotonically increasing function.

Variance Preserving SDE (DDPM)

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

$$\mathbf{f}(\mathbf{x}, t) = -\frac{1}{2}\beta(t)\mathbf{x}(t), \quad g(t) = \sqrt{\beta(t)}$$

Variance is preserved if $\mathbf{x}(0)$ has a unit variance.

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

Summary

- Continuous-in-time NF uses neural ODE to define continuous dynamic $\mathbf{z}(t)$. It has less functional restrictions.
- Kolmogorov-Fokker-Planck theorem allows to calculate $\log p(\mathbf{z}, t)$ at arbitrary moment t.
- FFJORD model makes such kind of NF scalable.
- Adjoint method generalizes backpropagation procedure and allows to train Neural ODE solving ODE for adjoint function back in time.
- 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.
- There exists special probability flow ODE for each SDE that gives the same probability path.
- It is possible to revert SDE using score function.
- Score matching (NCSN) and diffusion models (DDPM) are the discretizations of the SDEs (variance exploding and