Deep Generative Models

Lecture 13

Roman Isachenko

Ozon Masters

Spring, 2022

Outline

1. Neural ODE: finish
2. Continuous-in-time normalizing flows
3. Langevin dynamic
4. Score matching

Outline

1. Neural ODE: finish
2. Continuous-in-time normalizing flows
3. Langevin dynamic
4. Score matching

Neural ODE

Adjoint functions

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

Theorem (Pontryagin)

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

Do we know any initilal condition?

Solution for adjoint function

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

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

Note: These equations are solved back in time.

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

Neural ODE

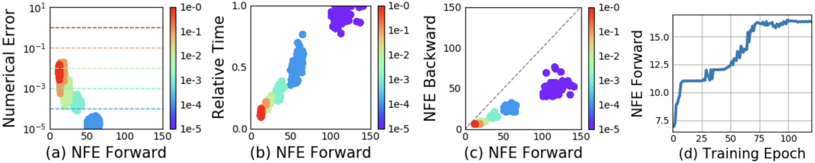
Forward pass

$$\mathbf{z}(t_1) = \int_{t_0}^{t_1} f(\mathbf{z}(t), \boldsymbol{\theta}) dt + \mathbf{z}_0 \quad \Rightarrow \quad \text{ODE Solver}$$

Backward pass

$$\left.\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\theta}(t_0)} &= \mathbf{a}_{\boldsymbol{\theta}}(t_0) = -\int_{t_1}^{t_0} \mathbf{a}_{\mathbf{z}}(t)^T \frac{\partial f(\mathbf{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}(t)} dt + 0 \\
\frac{\partial L}{\partial \mathbf{z}(t_0)} &= \mathbf{a}_{\mathbf{z}}(t_0) = -\int_{t_1}^{t_0} \mathbf{a}_{\mathbf{z}}(t)^T \frac{\partial f(\mathbf{z}(t), \boldsymbol{\theta})}{\partial \mathbf{z}(t)} dt + \frac{\partial L}{\partial \mathbf{z}(t_1)} \\
\mathbf{z}(t_0) &= -\int_{t_1}^{t_0} f(\mathbf{z}(t), \boldsymbol{\theta}) dt + \mathbf{z}_1.
\end{aligned}\right\} \Rightarrow \text{ODE Solver}$$

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. Neural ODE: finish
2. Continuous-in-time normalizing flows
3. Langevin dynamic
4. Score matching

Continuous Normalizing Flows

Discrete Normalizing Flows

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

Continuous-in-time dynamic transformation

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

Assume that function f is uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t. From Picard's existence theorem, it follows that the above ODE has a **unique solution**.

Forward and inverse transforms

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

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

Papamakarios G. et al. Normalizing flows for probabilistic modeling and inference, 2019

Continuous Normalizing Flows

To train this flow we have to get the way to calculate the density $p(\mathbf{z}(t))$.

Theorem (special case of Kolmogorov-Fokker-Planck)

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

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

Note: Unlike discrete-in-time flows, the function f does not need to be bijective, because uniqueness guarantees that the entire transformation is automatically bijective.

Density evaluation

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

Adjoint method is used to integral evaluation.

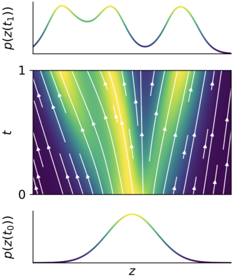
Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

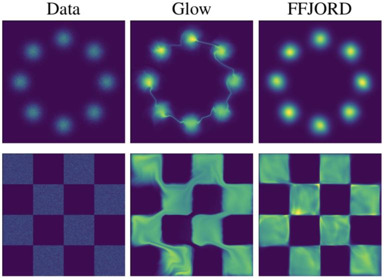
Continuous Normalizing Flows

Forward transform + log-density

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

- Discrete-in-time normalizing flows need invertible f. It costs $O(m^3)$ to get determinant of the Jacobian.
- Continuous-in-time flows require only smoothness of f. It costs $O(m^2)$ to get the trace of the Jacobian.





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

Continuous Normalizing Flows

- $\text{tr}\left(\frac{\partial f(\mathbf{z}(t), \boldsymbol{\theta})}{\partial \mathbf{z}(t)}\right)$ costs $O(m^2)$, or approximately the same cost as m evaluations of f, since each entry of the diagonal of the Jacobian requires computing a separate derivative of f.
- Vector-Jacobian products $\mathbf{v}^T \frac{\partial f}{\partial \mathbf{z}}$ can be computed for approximately the same cost as evaluating f.

It is possible to reduce cost from $O(m^2)$ to $O(m)$!

Hutchinson's trace estimator

$$\text{tr}(A) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T A \boldsymbol{\epsilon}\right]; \quad \mathbb{E}[\boldsymbol{\epsilon}] = 0; \quad \text{Cov}(\boldsymbol{\epsilon}) = I.$$

FFJORD density estimation

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

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

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

Outline

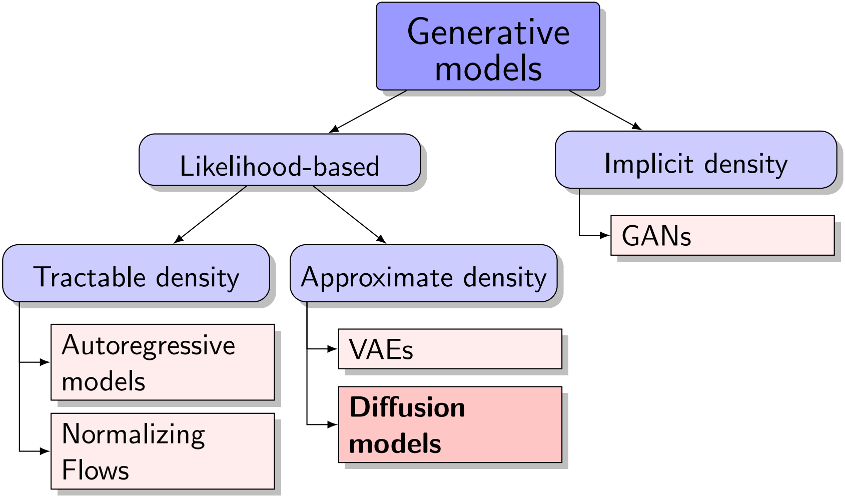
1. Neural ODE: finish

2. Continuous-in-time normalizing flows

3. Langevin dynamic

4. Score matching

Generative models zoo



Langevin dynamic

Imagine that we have some generative model $p(\mathbf{x}|\boldsymbol{\theta})$.

Statement

Let $\mathbf{x}_0$ be a random vector. Then under mild regularity conditions for small enough η samples from the following dynamics

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{1}{2} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta}) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1).$$

will comes from $p(\mathbf{x}|\boldsymbol{\theta})$.

What do we get if $\boldsymbol{\epsilon} = \mathbf{0}$?

Energy-based model

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{\hat{p}(\mathbf{x}|\boldsymbol{\theta})}{Z_{\boldsymbol{\theta}}}, \quad \text{where } Z_\theta = \int \hat{p}(\mathbf{x}|\boldsymbol{\theta}) d\mathbf{x}$$

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\boldsymbol{\theta}) = \nabla_{\mathbf{x}} \log \hat{p}(\mathbf{x}|\boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log Z_{\boldsymbol{\theta}} = \nabla_{\mathbf{x}} \log \hat{p}(\mathbf{x}|\boldsymbol{\theta})$$

Stochastic differential equation (SDE)

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

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

- $\mathbf{w}(t)$ is the standard Wiener process (Brownian motion)

  $\mathbf{w}(t) - \mathbf{w}(s) \sim \mathcal{N}(0, t - s), \quad d\mathbf{w} = \epsilon \cdot \sqrt{dt}$, where $\epsilon \sim \mathcal{N}(0, 1)$.
- $\mathbf{f}(\mathbf{x}, t)$ is the **drift** function of $\mathbf{x}(t)$.
- $g(t)$ is the **diffusion** coefficient of $\mathbf{x}(t)$.
- If $g(t) = 0$ we get standard ODE.

How to get distribution $p(\mathbf{x}|t)$ for $\mathbf{x}(t)$?

Stochastic differential equation (SDE)

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

Theorem (Kolmogorov-Fokker-Planck)

Evolution of the distribution $p(\mathbf{x}|t)$ is given by the folliwing ODE:

$$\frac{\partial p(\mathbf{x}|t)}{\partial t} = -\frac{\partial}{\partial \mathbf{x}}\big(\mathbf{f}(\mathbf{x}, t)p(\mathbf{x})\big) + \frac{1}{2}g^2(t)\frac{\partial^2 p(\mathbf{x}|t)}{\partial \mathbf{x}^2}$$

Langevin SDE

Let consider special case of SDE with $g(t) = 1$ and $\mathbf{f}(\mathbf{x}, t) = \frac{1}{2}\frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}|t)$.

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

Let apply KFP theorem.

Stochastic differential equation (SDE)

Langevin dynamic

Let discretize the Langevin SDE

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{1}{2} \frac{\partial}{\partial \mathbf{x}} \log p(\mathbf{x}|t) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1).$$

Statement

Let $\mathbf{x}_0$ be a random vector. Then under mild regularity conditions for small enough η samples from the following dynamics

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \frac{1}{2} \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\boldsymbol{\theta}) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1).$$

will comes from $p(\mathbf{x}|\boldsymbol{\theta})$.

The desity $p(\mathbf{x}|\boldsymbol{\theta})$ is a **stationary** distribution for this SDE.

Outline

1. Neural ODE: finish
2. Continuous-in-time normalizing flows
3. Langevin dynamic
4. Score matching

Score matching

We could sample from the model if we have $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\boldsymbol{\theta})$.

Fisher divergence

$$D_F(\pi, p) = \frac{1}{2}\mathbb{E}_{\pi}\bigl\|\nabla_{\mathbf{x}} \log p(\mathbf{x}|\boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x})\bigr\|_2^2 \rightarrow \min_{\boldsymbol{\theta}}$$

Score function

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

Problem: we do not know $\nabla_{\mathbf{x}} \log \pi(\mathbf{x})$.

Theorem

$$\frac{1}{2}\mathbb{E}_{\pi}\bigl\|\mathbf{s}(\mathbf{x}, \boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x})\bigr\|_2^2 = \mathbb{E}_{\pi}\Bigl[\frac{1}{2}\|\mathbf{s}(\mathbf{x}, \boldsymbol{\theta})\|_2^2 + \text{tr}\bigl(\nabla_{\mathbf{x}}\mathbf{s}(\mathbf{x}, \boldsymbol{\theta})\bigr)\Bigr] + \text{const}$$

Here $\nabla_{\mathbf{x}}\mathbf{s}(\mathbf{x}, \boldsymbol{\theta}) = \nabla^2_{\mathbf{x}} \log p(\mathbf{x}|\boldsymbol{\theta})$ is a Hessian matrix.

Hyvarinen A. Estimation of non-normalized statistical models by score matching, 2005

Score matching

Theorem

$$\frac{1}{2}\mathbb{E}_{\pi}\Big\|\mathbf{s}(\mathbf{x},\boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x})\Big\|_2^2 = \mathbb{E}_{\pi}\left[\frac{1}{2}\|\mathbf{s}(\mathbf{x},\boldsymbol{\theta})\|_2^2 + \mathrm{tr}\big(\nabla_{\mathbf{x}}\mathbf{s}(\mathbf{x},\boldsymbol{\theta})\big)\right] + \text{const}$$

Proof (only for 1D)

$$\mathbb{E}_{\pi}\Big\|s(x) - \nabla_x \log \pi(x)\Big\|_2^2 = \mathbb{E}_{\pi}\left[s(x)^2 + (\nabla_x \log \pi(x))^2 - 2[s(x)\nabla_x \log \pi(x)]\right]$$

$$\begin{aligned}
\mathbb{E}_{\pi}[s(x)\nabla_x \log \pi(x)] &= \int \pi(x)\nabla_x \log p(x)\nabla_x \log \pi(x) dx \\
&= \int \nabla_x \log p(x)\nabla_x \pi(x) dx = \pi(x)\nabla_x \log p(x)\Big|_{-\infty}^{+\infty} \\
&= -\int \nabla_x^2 \log p(x)\pi(x) dx = -\mathbb{E}_{\pi}\nabla_x^2 \log p(x)
\end{aligned}$$

$$\frac{1}{2}\mathbb{E}_{\pi}\Big\|s(x) - \nabla_x \log \pi(x)\Big\|_2^2 = \frac{1}{2}\mathbb{E}_{\pi}\Big[s(x)^2 + \nabla_x s(x)\Big] + \text{const.}$$

Hyvarinen A. Estimation of non-normalized statistical models by score matching, 2005

Summary

- Adjoint method generalizes backpropagation procedure and allows to train Neural ODE solving ODE for adjoint function back in time.

- Kolmogorov-Fokker-Planck theorem allows to construct continuous-in-time normalizing flow with less functional restrictions.

- FFJORD model makes such kind of flows scalable.