Deep Generative Models

Lecture 13

Roman Isachenko

Ozon Masters

Spring, 2022

Outline

1. Neural ODE: finish
2. Continuous-in-time normalizing flows
3. Diffusion Models

Outline

1. Neural ODE: finish
2. Continuous-in-time normalizing flows
3. Diffusion Models

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

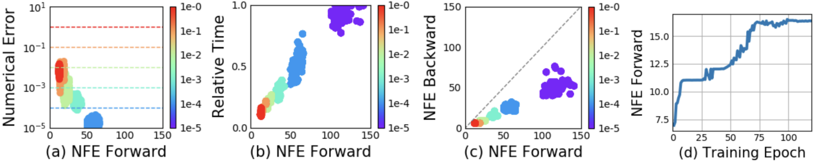
Forward pass

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

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

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 (Fokker-Planck)

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

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\text{trace}\left(\frac{\partial f(\mathbf{z}(t), \boldsymbol{\theta})}{\partial \mathbf{z}(t)}\right).$$

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

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

Adjoint method is used to integral evaluation.

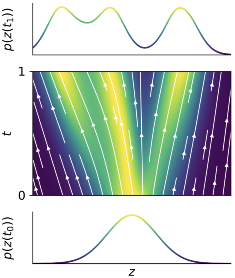
Chen R. T. Q. et al. Neural Ordinary Differential Equations, 2018

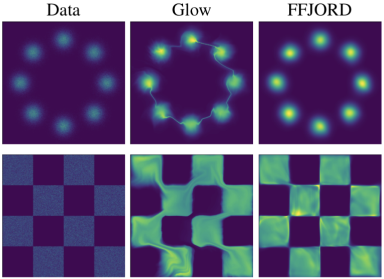
Continuous Normalizing Flows

Forward transform + log-density

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

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





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. Diffusion Models

Summary

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

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

- FFJORD model makes such kind of flows scalable.