

Deep Generative Models

Lecture 13

Roman Isachenko



Spring, 2022

Outline

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

Recap of previous lecture

Discrete VAE latents

- ▶ Define dictionary (word book) space $\{\mathbf{e}_k\}_{k=1}^K$, where $\mathbf{e}_k \in \mathbb{R}^C$, K is the size of the dictionary.
- ▶ Our variational posterior $q(c|\mathbf{x}, \phi) = \text{Categorical}(\pi(\mathbf{x}, \phi))$ (encoder) outputs discrete probabilities vector.
- ▶ We sample c^* from $q(c|\mathbf{x}, \phi)$ (reparametrization trick analogue).
- ▶ Our generative distribution $p(\mathbf{x}|\mathbf{e}_{c^*}, \theta)$ (decoder).

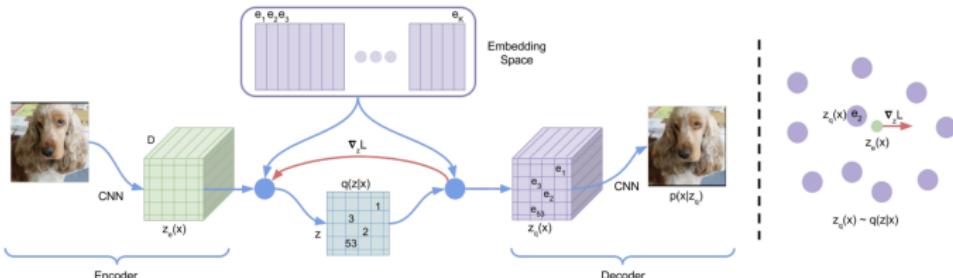
ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|c, \theta) - KL(q(c|\mathbf{x}, \phi) || p(c)) \rightarrow \max_{\phi, \theta} .$$

KL term

$$KL(q(c|\mathbf{x}, \phi) || p(c)) = -H(q(c|\mathbf{x}, \phi)) + \log K.$$

Recap of previous lecture



Deterministic variational posterior

$$q(c_{ij} = k^* | \mathbf{x}, \phi) = \begin{cases} 1, & \text{for } k^* = \arg \min_k \|[\mathbf{z}_e]_{ij} - \mathbf{e}_k\|; \\ 0, & \text{otherwise.} \end{cases}$$

ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{e}_c, \theta) - \log K = \log p(\mathbf{x}|\mathbf{z}_q, \theta) - \log K.$$

Straight-through gradient estimation

$$\frac{\partial \log p(\mathbf{x}|\mathbf{z}_q, \theta)}{\partial \phi} = \frac{\partial \log p(\mathbf{x}|\mathbf{z}_q, \theta)}{\partial \mathbf{z}_q} \cdot \frac{\partial \mathbf{z}_q}{\partial \phi} \approx \frac{\partial \log p(\mathbf{x}|\mathbf{z}_q, \theta)}{\partial \mathbf{z}_q} \cdot \frac{\partial \mathbf{z}_e}{\partial \phi}$$

Recap of previous lecture

Gumbel-max trick

Let $g_k \sim \text{Gumbel}(0, 1)$ for $k = 1, \dots, K$. Then

$$c = \arg \max_k [\log \pi_k + g_k]$$

has a categorical distribution $c \sim \text{Categorical}(\pi)$.

Gumbel-softmax relaxation

Concrete distribution = continuous + discrete

$$\hat{c}_k = \frac{\exp\left(\frac{\log q(k|\mathbf{x}, \phi) + g_k}{\tau}\right)}{\sum_{j=1}^K \exp\left(\frac{\log q(j|\mathbf{x}, \phi) + g_j}{\tau}\right)}, \quad k = 1, \dots, K.$$

Reparametrization trick

$$\nabla_\phi \mathbb{E}_{q(c|\mathbf{x}, \phi)} \log p(\mathbf{x}|\mathbf{e}_c, \theta) = \mathbb{E}_{\text{Gumbel}(0,1)} \nabla_\phi \log p(\mathbf{x}|\mathbf{z}, \theta),$$

where $\mathbf{z} = \sum_{k=1}^K \hat{c}_k \mathbf{e}_k$ (all operations are differentiable now).

Maddison C. J., Mnih A., Teh Y. W. *The Concrete distribution: A continuous relaxation of discrete random variables*, 2016

Jang E., Gu S., Poole B. *Categorical reparameterization with Gumbel-Softmax*, 2016

Recap of previous lecture

Consider Ordinary Differential Equation

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

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

Euler update step

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

Residual block

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t, \theta)$$

It is equivalent to Euler update step for solving ODE with $\Delta t = 1$!

In the limit of adding more layers and taking smaller steps we get:

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \theta); \quad \mathbf{z}(t_0) = \mathbf{x}; \quad \mathbf{z}(t_1) = \mathbf{y}.$$

Neural ODE

Forward pass (loss function)

$$\begin{aligned} L(\mathbf{y}) &= L(\mathbf{z}(t_1)) = L \left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), \theta) dt \right) \\ &= L(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)) \end{aligned}$$

Note: ODESolve could be any method (Euler step, Runge-Kutta methods).

Backward pass (gradients computation)

For fitting parameters we need gradients:

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

In theory of optimal control these functions called **adjoint** functions. They show how the gradient of the loss depends on the hidden state $\mathbf{z}(t)$ and parameters θ .

Outline

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

Neural ODE

Adjoint functions

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

Theorem (Pontryagin)

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

Do we know any initial condition?

Solution for adjoint function

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

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

Note: These equations are solved back in time.

Neural ODE

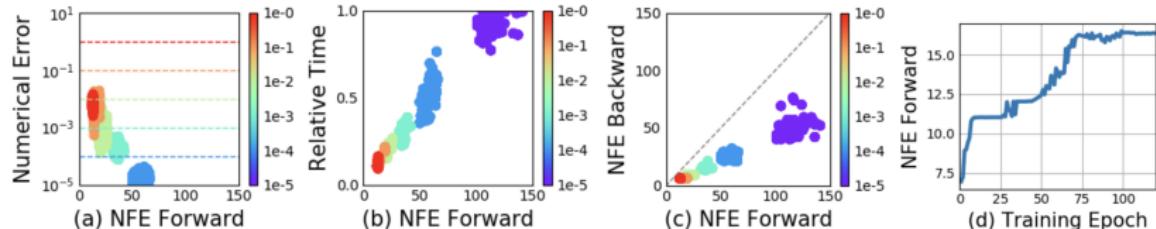
Forward pass

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

Backward pass

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

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



Outline

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

Continuous Normalizing Flows

Discrete Normalizing Flows

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

Continuous-in-time dynamics

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

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

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

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

Continuous Normalizing Flows

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

Theorem (special case of Kolmogorov-Fokker-Planck)

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

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

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

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

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

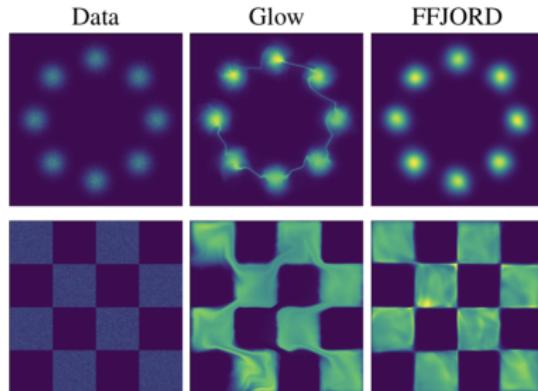
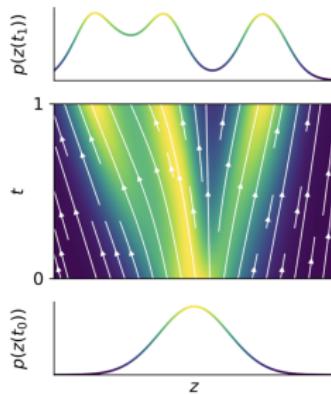
Adjoint method is used for getting the derivatives.

Continuous Normalizing Flows

Forward transform + log-density

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

- 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

- ▶ $\text{tr} \left(\frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} \right)$ costs $O(m^2)$ (m evaluations of f), since we have to compute a derivative for each diagonal element.
- ▶ Jacobian vector 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

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

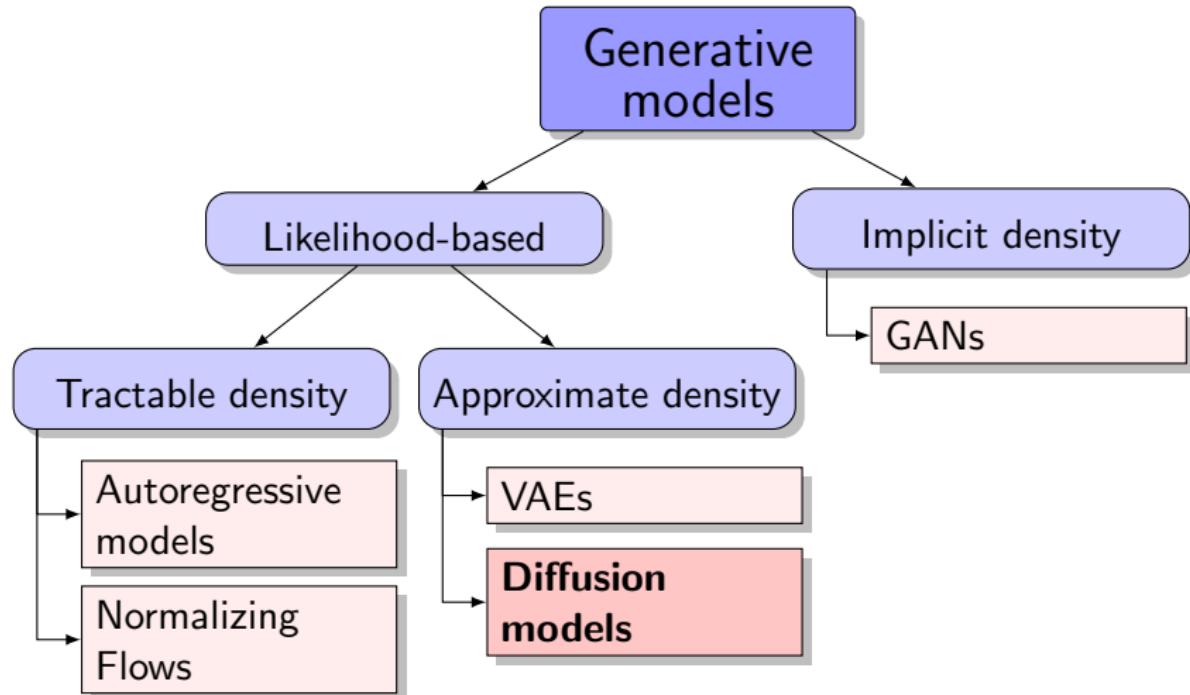
FFJORD density estimation

$$\begin{aligned} \log p(\mathbf{z}(t_1)) &= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \text{tr} \left(\frac{\partial f(\mathbf{z}(t), \theta)}{\partial \mathbf{z}(t)} \right) dt = \\ &= \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\epsilon)} \int_{t_0}^{t_1} \left[\epsilon^T \frac{\partial f}{\partial \mathbf{z}} \epsilon \right] dt. \end{aligned}$$

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

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

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

Stochastic differential equation (SDE)

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

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

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

$$\mathbf{w}(t) - \mathbf{w}(s) \sim \mathcal{N}(0, t-s), \quad d\mathbf{w} = \epsilon \cdot \sqrt{dt}, \text{ where } \epsilon \sim \mathcal{N}(0, 1).$$

- ▶ $\mathbf{f}(\mathbf{x}, t)$ 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 $\mathbf{x}(t)$?

Theorem (Kolmogorov-Fokker-Planck)

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

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

Stochastic differential equation (SDE)

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

Langevin SDE (special case)

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

Langevin discrete dynamic

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

Let apply KFP theorem.

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

The density $p(\mathbf{x}, t) = \text{const.}$

Stochastic differential equation (SDE)

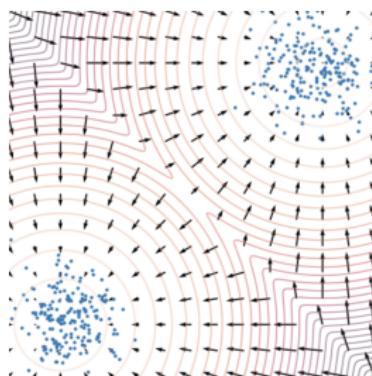
Statement

Let \mathbf{x}_0 be a random vector. Then 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 come from $p(\mathbf{x} | \boldsymbol{\theta})$ under mild regularity conditions for small enough η and large enough t .

The density $p(\mathbf{x} | \boldsymbol{\theta})$ 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 $\nabla_{\mathbf{x}} \log p(\mathbf{x}|\theta)$.

Fisher divergence

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

Score function

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

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

Theorem

Under some regularity conditions, it holds

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

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

Score matching

Theorem

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

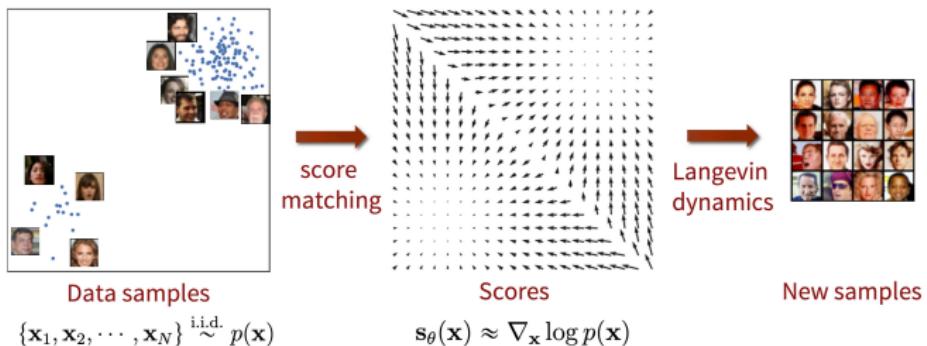
Proof (only for 1D)

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

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

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

Score matching



Theorem

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

1. The left hand side is intractable due to unknown $\pi(\mathbf{x})$ – **denoising score matching**.
2. The right hand side is complex due to Hessian matrix – **sliced score matching**.

Score matching

Sliced score matching (Hutchinson's trace estimation)

$$\mathbb{E}_\pi \left[\text{tr}(\nabla_{\mathbf{x}} \mathbf{s}(\mathbf{x}, \theta)) \right] = \mathbb{E}_{\pi(\mathbf{x})} \mathbb{E}_{p(\epsilon)} \left[\epsilon^T \nabla_{\mathbf{x}} \mathbf{s}(\mathbf{x}, \theta) \epsilon \right],$$

where $\mathbb{E}[\epsilon] = 0$ and $\text{Cov}(\epsilon) = I$.

Denoising score matching

Let perturb original data by normal noise $p(\mathbf{x}|\mathbf{x}', \sigma) = \mathcal{N}(\mathbf{x}|\mathbf{x}', \sigma^2 \mathbf{I})$

$$\pi(\mathbf{x}|\sigma) = \int \pi(\mathbf{x}') p(\mathbf{x}|\mathbf{x}', \sigma) d\mathbf{x}'.$$

Then the solution of

$$\frac{1}{2} \mathbb{E}_{\pi(\mathbf{x}|\sigma)} \| \mathbf{s}(\mathbf{x}, \theta, \sigma) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x}|\sigma) \|_2^2 \rightarrow \min_{\theta}$$

satisfies $\mathbf{s}(\mathbf{x}, \theta, \sigma) \approx \mathbf{s}(\mathbf{x}, \theta, 0) = \mathbf{s}(\mathbf{x}, \theta)$ using small enough noise scale σ .

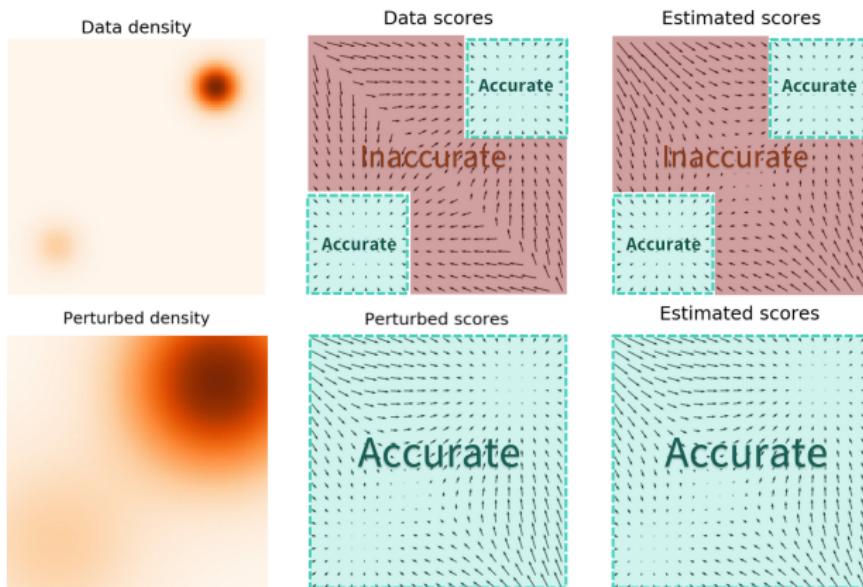
Song Y. Sliced Score Matching: A Scalable Approach to Density and Score Estimation, 2019

Vincent P. A connection between score matching and denoising autoencoders. Neural computation, 2011

Denoising score matching

Theorem

$$\begin{aligned}\mathbb{E}_{\pi(\mathbf{x}|\sigma)} \|\mathbf{s}(\mathbf{x}, \theta, \sigma) - \nabla_{\mathbf{x}} \log \pi(\mathbf{x}|\sigma)\|_2^2 = \\ = \mathbb{E}_{\pi(\mathbf{x}')} \mathbb{E}_{p(\mathbf{x}|\mathbf{x}', \sigma)} \|\mathbf{s}(\mathbf{x}, \theta, \sigma) - \nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{x}', \sigma)\|_2^2\end{aligned}$$



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.
- ▶ Langevin dynamics allows to sample from the model using the score function (due to existence of stationary distribution for SDE).
- ▶ Score matching proposes to minimize Fisher divergence to get score function.
- ▶ Sliced score matching and denoised score matching are two techniques to get scalable algorithm for fitting Fisher divergence.