COMP 7070 Advanced Topics in AI and ML March 26, 2024

Lecture 11: Flow: how to derive Flow model

Instructor: Yifan Chen *Scribes: Wenbo Shang, Rui Cao* *Proof reader: Zhanke Zhou*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

11.1 Economics Explanation of the Dual Form for Optimal Transport

Neural ODE can also be applied to the economic field, and here we can present an economic explanation of the dual form of OT. The primal concept is the transport cost in the form of $\int c(x,y)d\pi(x,y)$, where $\pi(x,y)$ is coupling function, as shown in Figure 11.1.

The dual form is to outsource the task to a vendor, which means we have a collection fee including two fees: $\int_X \psi(x)d\mu(x)+\int_Y \psi(y)d\nu(y)$. The vendor will guarantee the condition $\varphi(x)+\psi(y) \leq c(x,y)$, where $c(x,y)$ denotes the cost that transport by yourself and the LHS is the transport fee. During the implication, we can represent as follows:

$$\int \psi(x)d\pi(x,y) + \int \psi(y)d\pi(x,y) = \int_x \psi(x)d\mu(x) + \int_y \psi(y)d\nu(y) \leq \int c(x,y)d\pi(x,y). \tag{11.1}$$

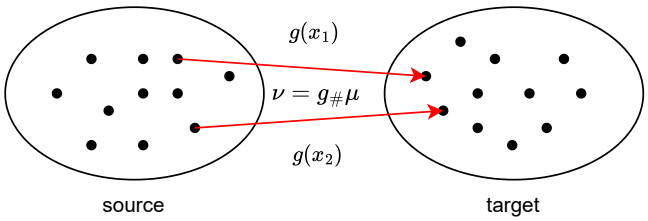

Figure 11.1: Transport cost

11.2 Preliminary of the Flow model

In this section, we focus on how to derive the flow model. Let's start with MLE first.

11.2.1 MLE

Maximum Likelihood Estimation (MLE) is a method used in statistics to estimate the parameters of a statistical model. It involves finding the parameter values that maximize the likelihood function, representing the probability of observing the given data under the specified model. Here we first present the formula of the distribution of variable x and its MLE as below:

$$q_\theta(x) = \int q(z)q(x|z)\,dx. \tag{11.2}$$

$$MLE = \max_\theta \mathop{\mathbb{E}}_{x\sim p(x)} \log q_\theta(x). \tag{11.3}$$

VAE formally addresses the MLE issue using ELBO while GAN leverages discriminator and minimax game. However, the Flow model directly does integration with Equation 11.2.

11.2.2 Settings of Flow model

Assume the variable z satisfies $\mu(z) \sim \mathcal{N}(0, \mathrm{I})$, to remove randomness from variable x, we can represent $q(x|z)$ as follows:

$$q(x|z) = \delta(x - g(z)). \tag{11.4}$$

where g is a generator. Accordingly, we can equivalently get the following formula:

$$q_\theta(x) = \int \delta(x - g(z))\, d\mu(z). \tag{11.5}$$

Thus, we have $q_\theta \sim g_\# \mu$ where $g_\# \mu$ represents a new distribution that given map g, $g_\# \mu$ can be formed from the original one μ. In other words, we have $\mu \sim g_\#^{-1} q_\theta$. Here we can derive the target density function:

$$q_\theta(x) = \mu(g^{-1}(x)) \cdot \left| \frac{Dz}{Dx} \right|. \tag{11.6}$$

where g is invertible and Jacobin determinant $\left|\frac{Dz}{Dx}\right|$ is easy to compute.

11.3 NICE

Nice is the first flow model paper that made the Jacobin matrix into a triangle matrix. Specifically, assume matrix h can be embedded into two pairs like this:

$$h = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ x_2 + m_\theta(x_1) \end{pmatrix}. \tag{11.7}$$

Thus, we can obtain:

$$\frac{Dh}{DX} = \begin{pmatrix} I & \\ \frac{Dm_\sigma}{DX_1} & I \end{pmatrix} \Rightarrow |\frac{Dh}{DX}| = 1. \tag{11.8}$$

We can also compute the convertible process:

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 - m_0(X_1) \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 - w_0(h_1) \end{pmatrix} = g^{-1}(h). \tag{11.9}$$

11.3.1 How to make the setting useful

First,

$$h_l \circ h_{l_{-1}} \cdots \circ h_0(x) = z. \tag{11.10}$$

where z is noise. Second, shuffle h_1 and h_2. Third, scale transforms. Then, we can derive the following equation from the last layer:

$$z = S \odot h_e, \Rightarrow \frac{Dz}{Dh_e} = diag(S). \tag{11.11}$$

where $\odot$ is the Hadamard product.

11.3.2 Real-valued non-volume preserving transformation

We know that

$$\left| \frac{Dz}{Dh_e} \right| \neq 1. \tag{11.12}$$

So, we can derive the final form

$$h = \begin{pmatrix} x_1 \\ S(x_1) \odot x_2 + m(x_1) \end{pmatrix} \Rightarrow \frac{Dh}{DX} = \begin{pmatrix} I & \\ \frac{DS}{DX_1} \odot X_1 & diag(S) \end{pmatrix}. \tag{11.13}$$

where $S(x_1) = exp(\log S)$ and $\log S = \mathrm{NN}(x_1)$. A positive S can be generated by equation $S(x_1) = exp(\log S)$. In this way, we can easily compute the Jacobin determinant.

11.3.3 How to use convolution as a trick

First, we can shuffle the channel but not X, Y. Second, we can use squeezing to increase channel size. For example,

$$h \times w \times c \to \frac{h}{2} \times \frac{w}{2} \times 4c. \tag{11.14}$$

11.3.4 How to sample after obtaining the model

Assume $z \sim \mathcal{N}(0, \mathbf{I})$, we can sample $z = h_l$, with $g_0^{-1} \circ g_1^{-1} \circ \cdots g_l^{-1}(z)$.

11.3.5 Multi-level

Given matrix X, we can design multi-level flows:

$$X \xrightarrow{flow_1} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \xrightarrow{flow_2} \begin{pmatrix} z_3 \\ z_4 \end{pmatrix} \xrightarrow{flow_3} \begin{pmatrix} z_1 \\ z_3 \\ z_5 \end{pmatrix}$$

Notably, we cannot assume $\begin{pmatrix} z_1 \\ z_3 \\ z_5 \end{pmatrix} \sim N(0, \mathbf{I})$.

We have

$$\begin{aligned} P(z_1, z_3, z_5) &= P(z_1|z_3, z_5) \cdot P(z_3|z_5) \cdot P(z_5) \\ &= P(z_1|z_2) \cdot P(z_3|z_4) \cdot P(z_5). \end{aligned} \tag{11.15}$$

where $\sigma(z_5) = \sigma(z_4)$, $\sigma(z_3, z_5) = \sigma(z_2)$, $z_1 \sim \mathcal{N}(\mu(z_2), \Sigma(z_2))$, $z_3 \sim \mathcal{N}(\mu(z_4), \Sigma(z_4))$ and $z_5 \sim \mathcal{N}(\mu, \Sigma)$. And μ and Σ are tunable. Thus , we can regard Equation 11.15 as:

$$\begin{aligned} P(z_1, z_3, z_5) &= P(z_1|z_2) \cdot P(z_3|z_4) \cdot P(z_5), \\ &= \mathcal{N}(\mu(z_2), \Sigma(z_2)) \cdot \mathcal{N}(\mu(z_4), \Sigma(z_4)) \cdot \mathcal{N}(\mu, \Sigma). \end{aligned} \tag{11.16}$$

Here is the generation process:

$$\begin{aligned} &z_5 \sim \mathcal{N}(\mu, \Sigma), z_4 = flow_3^{-1}(z_5), \\ &z_3 \sim \mathcal{N}(\mu(z_4), \Sigma(z_4)), z_2 = flow_2^{-1}\begin{pmatrix} z_3, \\ z_4 \end{pmatrix}, \\ &z_1 \sim \mathcal{N}(\mu(z_1), \Sigma(z_2)), \\ &X = flow_1^{-1}\begin{pmatrix} z_1 \\ z_2 \end{pmatrix}. \end{aligned} \tag{11.17}$$

11.4 GLOW: Improve Shuffling

Here we first present the LU decomposition: $W = PLU$, where P is a permutation matrix, L is the lower triangle matrix, and U is the upper triangle matrix. Thus we have $|W| = |U| \cdot |L|$. However, Not all W can be represented by LU. Consider a W with $W_{11} = 0$, but still W is full rank. We derive that

$$\begin{aligned} W_{11} = L_{11} \cdot U_{11} = 0 &\implies L_{11} = 0 \; or \; U_{11} = 0 \\ &\implies L \; or \; U \; will \; not \; be \; full \; rank. \\ &\implies W = LU \; is \; not \; full \; rank. \end{aligned} \tag{11.18}$$

Here, the conclusion contradicts our original assumption. So, we need to use P.

11.5 Other tools

We also have other tools, which should be easy to compute for g_{-1} and $|\frac{Dz}{DX}|$. For example, we present Planner Flow here. Specifically, $f(\vec{x}) = \vec{x} + \vec{u} \cdot h(w^\top + b)$, where h is a scalar function. Accordingly, we have $|\frac{Df}{DX}| = 1 + u^\top w \cdot h'(w^\top + b)$.

11.6 Neural ODE

Let's start with the ResNet, here given the form of ResNet:

$$z_{l+1} = z_l + f(z_l; w_l). \tag{11.19}$$

In the ResNet, we usually meet the initial value problem, *i.e.*, z_0 and T_0. Thus, the solution to the equation is approximately equal to the solution of another equation:

$$\frac{\partial z}{\partial t} = NN(z(t), t; \theta). \tag{11.20}$$

where $z_{e+1} - z_e \approx \frac{\partial z}{\partial t}$. The inference equation of the ODE can be:

$$z_0 + \int_{T_0}^{T_1} NN(z(t), t; \theta) = z(T_1). \tag{11.21}$$

Then we can reformula the training process of the ResNet. We can apply an ODE solver to the Equation 11.21 and obtain:

$$L(z(T_1)) = L(ODE\ Solver(z(T_0), NN_\theta, T_0, T_1, \theta)). \tag{11.22}$$

However, people may worry about whether the method is feasible for PyTorch. For inference, we only need to store NN_θ in the GPU memory(*e.g.*, L layers for ResNet). Besides, directly using Autodiff in PyTorch will cause $\mathcal{O}(L)$ memory and we can use Adjoint Sensitivity Method instead.

Suppose the adjoint state $a(t) = \frac{dL}{dz(t)}$; we can obtain the following equation after complex derivation.

$$\frac{da}{dt} = -a(t) \cdot \frac{\partial f(z(t), t; \theta)}{\partial z(t)}. \tag{11.23}$$

Thus we can put the Equation 11.27 into the equation below.

$$a(T_1) = \frac{dL}{dz(T_1)} \Rightarrow a(T_0) = a(T_1) + \int_{T_1}^{T_0} \frac{da}{dt} dt. \tag{11.24}$$

Since our target is to calculate $\frac{dL}{d\theta}$, we need to achieve $a(T_1) = \frac{\partial L}{\partial z}$ after operations on $z(T_0)$ and $z(T_1)$.

$$\begin{aligned} a_{aug} &= (a, a_\theta = \frac{dL}{d\theta}, a_t = \frac{dL}{dt}), \\ f_{aug} &= (f, \theta, t). \end{aligned} \tag{11.25}$$

Then it yields

$$\frac{Df_{aug}}{D(z, \theta, t)} = \begin{pmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial \theta} & \frac{\partial f}{\partial t} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{11.26}$$

For $\frac{da}{dt}$, we can achieve

$$\begin{aligned} \frac{da}{dt} &= -a(t) \cdot \frac{\partial f}{\partial t}, \\ \frac{da_\theta}{dt} &= -a_\theta(t) \cdot \frac{\partial f}{\partial \theta}. \end{aligned} \tag{11.27}$$

Specifically, we know

$$a(t) = \frac{dL}{dz(t)} = \frac{dL}{dz(t+\varepsilon)} \cdot \frac{dz(t+\varepsilon)}{dz(t)}. \tag{11.28}$$

where $z(t+\varepsilon) = a(t+\varepsilon)$, L is influenced by $z(t+\varepsilon)$ and $z(t+\varepsilon)$ is decided by $z(t)$. Thus we have

$$\begin{aligned}
\frac{da}{dt} &= \lim_{\varepsilon \to 0} \frac{a(t+\varepsilon) - a(t)}{\varepsilon} = \lim_{\varepsilon \to 0^+} \frac{a(t+\varepsilon) - a(t)}{\varepsilon}, \\
&= \lim_{\varepsilon \to 0^+} \frac{a(t+\varepsilon) - a(t+\varepsilon)\frac{d(z(t)+\int_t^{t+\varepsilon} f(z(s),s;\theta)ds)}{dz(t)}}{\varepsilon}, \\
&= \lim_{\varepsilon \to 0^+} -\frac{a(t+\varepsilon)}{\varepsilon} \cdot \frac{\int_t^{t+\varepsilon} f(z(s),s;\theta)ds}{dz(t)}, \\
&= \lim_{\varepsilon \to 0^+} -\frac{a(t+\varepsilon)}{\varepsilon} \cdot \frac{\varepsilon \cdot \partial f(z(t),t;\theta)}{\partial z(t)}, \\
&= -a(t) \cdot \frac{\partial f(z(t),t;\theta)}{\partial z(t)}.
\end{aligned} \tag{11.29}$$

Thus we prove the Equation 11.27. Then we can obtain

$$a_\theta(T_0) \equiv \frac{dL}{d\theta(T_0)} = a_\theta(T_1) + \int_{T_1}^{T_0} \frac{da_\theta}{dt} dt. \tag{11.30}$$

where $a_\theta(T_1) = 0$ and $\int_{T_1}^{T_0} \frac{da_\theta}{dt} dt$ is our target.