Generative Flow

John Thickstun

Suppose p_θ is defined implicitly by the pushforward of a density q using a generator $g_\theta : \mathcal{Z} \to \mathcal{X}$. Changing variables from z to x, we see that

$$\Pr(A) = \Pr(g_\theta^{-1}(A)) = \int_{g_\theta^{-1}(A)} q(z)\, dz = \int_A q(g_\theta^{-1}(x))|\nabla_x g_\theta^{-1}(x)|\, dx. \tag{1}$$

And therefore the density $q(z)$ pushes forward to

$$p_\theta(x) = q(g_\theta^{-1}(x))|\nabla_x g_\theta^{-1}(x)|. \tag{2}$$

If the inverse of g_θ and its Jacobian are easily computed, we can use Equation (2) to convert a generator into a density estimator. Furthermore, we can train this generator using maximum likelihood estimation. Given samples $x_1, \ldots, x_n \sim p$,

$$\hat{\theta}_{\text{mle}} = \arg\min_\theta \mathop{\mathbb{E}}_{x\sim p} -\log p_\theta(x) \approx \arg\min_\theta \sum_{i=1}^n -\log q(g_\theta^{-1}(x_i)) - \operatorname{logdet} \nabla_x g_\theta^{-1}(x_i). \tag{3}$$

Generative Flow

We've already seen one example of a network architecture g_θ for which we can compute cheap inverses and Jacobians: inverse autoregressive flow (IAF) [Kingma et al., 2016]. These flows worked well for parameterizing variational posteriors, but they are not so suitable for parameterizing the pushforward distribution g_θ. The problem is that we need to be able to both compute $x = g_\theta(z) \sim p_\theta$ (sampling) and also compute inverses $z = g_\theta^{-1}(x)$ (inference). If we parameterize g_θ^{-1} with IAF, then sampling is an expensive sequence of autoregressive computation. Parameterizing g_θ with IAF creates similar problems: sampling becomes a cheap, parallel operation but inference is expensive. This dilemma is avoided for posterior approximation, because that task requires samples $z = g_\theta^{-1}(x)$ rather than samples $x = g_\theta(z)$. Therefore, by parameterizing g_θ^{-1} with IAF, both sampling and inference are cheap.

A clever idea that adapts flow to parameterizing a pushforward distribution is developed in Dinh et al. [2015]. The idea is to partition your features $x \in \mathcal{X}$ into two sets. E.g. if $\mathcal{X} = \mathbb{R}^d$ then we can partition into the first $d/2$ and second $d/2$ features: $x = (x_{1,\ldots,d/2}, x_{d/2+1,\ldots,d})$. We parameterize a function $h_\theta : \mathbb{R}^{d/2} \to \mathbb{R}^{d/2}$ using an expressive neural network, and proceed to construct an **additive coupling** $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ defined by the rule $x = g_\theta(z)$ where

$$x_{1,\ldots,d/2} = z_{1,\ldots,d/2}, \tag{4}$$

$$x_{d/2+1,\ldots,d} = z_{d/2+1,\ldots,d} + h_\theta(z_{1,\ldots,d/2}). \tag{5}$$

Observe that this coupling function is invertible, regardless of h_θ. Given $x \in \mathbb{R}^d$, we can recover $z = g_\theta^{-1}(x)$ by computing

$$z_{1,\dots,d/2} = x_{1,\dots,d/2}, \tag{6}$$

$$z_{d/2+1,\dots,d} = x_{d/2+1,\dots,d} - h_\theta(x_{1,\dots,d/2}). \tag{7}$$

And the Jacobian of the inverse transformation is simply

$$\nabla_x g_\theta^{-1}(x) = \begin{bmatrix} \mathrm{Id}_{d/2} & 0 \\ -\frac{\partial h_\theta(x_{1,\dots,d/2})}{\partial x_{1,\dots,d/2}} & \mathrm{Id}_{d/2} \end{bmatrix}.$$

Therefore $\mathrm{logdet}(\nabla_x g_\theta^{-1}(x)) = 0$: this is a volume-preserving transformation.

While an affine coupling is not very expressive (it is the identity on half of the features!) the idea is to chain multiple affine couplings $g_\theta^{(k)}$ together (with different partitions at each step) to construct a deep network $g_\theta(z) = g_\theta^{(L)} \circ \cdots \circ g_\theta^{(1)}(z)$. Curiously, these chains of additive couplings are the same construction as the Feistel network [Luby and Rackoff, 1988] used in the cryptography community to construct ciphers (e.g. DES and Blowfish). Even at significant depth, generative flow models seem to suffer empirically from this restrictive model structure, requiring substantially more parameters and training time than other modeling techniques (at least for popular vision tasks). Some (very preliminary) work that attempts to remove the architectural restrictions using approximations to the inverse and its Jacobian are presented in Keller et al. [2020].

One simple extension of additive flow is the real non-volume preserving transformation (real-NVP) [Dinh et al., 2017] that replaces the additive couplings (4) and (5) with affine couplings. Given parameterized scale and translation functions $s_\theta, t_\theta : \mathbb{R}^{d/2} \to \mathbb{R}^{d/2}$, we define an affine coupling by $x = g_\theta(z)$ by

$$x_{1,\dots,d/2} = z_{1,\dots,d/2}, \tag{8}$$

$$x_{d/2+1,\dots,d} = z_{d/2+1,\dots,d} \odot \exp(s_\theta(z_{1,\dots,d/2})) + t_\theta(z_{1,\dots,d/2}). \tag{9}$$

The multiplicative scaling $\odot$ should be interpreted elementwise. The Jacobian in this case is

$$\nabla_x g_\theta^{-1}(x) = \begin{bmatrix} \mathrm{Id}_{d/2} & 0 \\ \dots & \mathrm{diag}\left(\exp(-s_\theta(x_{1,\dots,d/2}))\right) \end{bmatrix} \tag{10}$$

And the log determinant of the Jacobian is just

$$\mathrm{logdet}(\nabla_x g_\theta^{-1}(x)) = -\sum_{i=1}^{d/2} s_\theta(x_{1,\dots,d/2})_i. \tag{11}$$

Extensions of this idea with large-scale experiments is presented in Kingma and Dhariwal [2018].

Continuous Flows

Let $\mathbf{z}_t = g_t \circ \cdots \circ g_1(\mathbf{z}_0)$. If q_t is the pushforward of $q_0 = q$ induced by g_t then

$$\log q_t(\mathbf{z}_t) = \log q_0(\mathbf{z}_0) - \sum_{s=1}^{t} \mathrm{logdet}\left(\frac{\partial g_s(\mathbf{z}_{s-1})}{\partial \mathbf{z}_{s-1}}\right). \tag{12}$$

We can think of the sequence of function applications $g_1, \ldots, g_t$ as gradually warping the initial distribution q_0 into the final distribution q_t. An interesting way to interpret an additive update like Equation (5) (especially as t gets large) is to think about it as an (Euler) discretization of continuous dynamics h_θ [Chen et al., 2018]; i.e.

$$\mathbf{z}_t = \int_0^t h_\theta(\mathbf{z}_s)\, ds \approx \mathbf{z}_0 + \sum_{s=1}^{t} h_\theta(\mathbf{z}_{s-1}). \tag{13}$$

The instantaneous change of variables under dynamics $\frac{\partial \mathbf{z}}{\partial t} = h_\theta(\mathbf{z})$, analogous to the discrete change Equation (2), are given by

$$\frac{\partial \log q_s(\mathbf{z}(s))}{\partial s} = -\operatorname{tr}\left(\frac{\partial h_\theta(\mathbf{z}_s)}{\partial \mathbf{z}}\right). \tag{14}$$

A proof of this fact can be found in Appendix A of Chen et al. [2018]. We can accumulate these instantaneous updates to the density, analogous to Equation (12), to compute

$$\log q_t(\mathbf{z}_t) = \log q_0(\mathbf{z}(0)) - \int_0^t \operatorname{tr}\left(\frac{\partial h_\theta(\mathbf{z}_s)}{\partial \mathbf{z}}\right) ds \tag{15}$$

Note that the (potentially expensive) determinant operation has been replaced by the (relatively inexpensive) trace operation in this analogy.

For the density estimation application, we can think of $x = \mathbf{z}_t$ and $\log p_\theta(x) = \log q_t(\mathbf{z}_t)$. For maximum likelihood estimation, we need to compute the sensitivity of $\log q_t(\mathbf{z}_t)$ to the dynamics parameters θ. This might look daunting, but the solution can be decomposed and characterized by introducing an adjoint process

$$\alpha(s) = \frac{\partial \log q_s(\mathbf{z}_s)}{\partial \mathbf{z}}. \tag{16}$$

This process can be characterized [Pontryagin, 1962] by the solution to another initial value problem with an initial value $\alpha(t) = \frac{\partial \log q_t(\mathbf{z}_t)}{\partial \mathbf{z}_t}$, running backward in time from $s = t$ to $s = 0$:

$$\frac{d\alpha(s)}{ds} = -\alpha(s)^T \frac{\partial h_\theta(\mathbf{z}_s)}{\partial \mathbf{z}}. \tag{17}$$

And combining solutions to Equation (13) and (17), we can write the gradient of the loss as the solution to the following integral:

$$\frac{d \log q_t(\mathbf{z}_t)}{d\theta} = \int_0^t \alpha(s)^T \frac{\partial h_\theta(\mathbf{z}_s)}{\partial \theta}\, ds. \tag{18}$$

For the results of this approach to training a generative flow model, see Grathwohl et al. [2018].

References

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, 2018. (document)

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *International Conference on Learning Representations Workshop*, 2015. (document)

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *International Conference on Learning Representations*, 2017. (document)

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2018. (document)

T Anderson Keller, Jorn WT Peters, Priyank Jaini, Emiel Hoogeboom, Patrick Forré, and Max Welling. Self normalizing flows. *Beyond Backpropagation Workshop at Neural Information Processing Systems*, 2020. (document)

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018. (document)

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, 2016. (document)

Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 1988. (document)

Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 1962. (document)