“Discrete Flows: Invertible Generative Models of Discrete Data” Notes

Feb 10

Summary:

1. A normalizing flow is a series of invertible and fully differentiable functions applied to a probability distribution, usually with the goal of approximating a high-dimensional distribution.
2. Traditionally normalizing flows apply only to continuous distributions, however, this paper formulates a generalization that applies to discrete distributions as well
3. This discrete normalizing flow is less computationally intense to calculate and more applicable to tasks where the underlying data is inherently discrete.

Background: Traditionally, a normalizing flow takes the following form. Let $\mathbf{x}$ be a D-dimensional continuous random variable. Given an invertible function, $f : \mathbb{R}^D \to \mathbb{R}^D$, $\mathbf{y} \equiv f(\mathbf{x})$:

$$p(\mathbf{y}) = p(f^{-1}(\mathbf{y})) \det \left| \frac{d\mathbf{x}}{d\mathbf{y}} \right| \tag{1}$$

The transformation f is referred to as a flow, and $p(\mathbf{x})$ is the base distribution. Multiple flows can be stacked on top of each other to increase the expressitivity of $p(\mathbf{y})$.

The determinant of this Jacobian incurs a complexity of $O(D^3)$ (naively) which is intractable for large datasets. Certain continuous normalizing flows are purposefully created so that the Jacobian in (1) is much more tractable, but they still are a computational burden.

Discrete Flows: Let $\mathbf{x}$ be a discrete random variable, and $\mathbf{y} \equiv f(\mathbf{x})$ where f is some function of x. The probability mass function of $\mathbf{y}$ is the sum over the "pre-image" of f:

$$p(\mathbf{y} = y) = \sum_{x \in f^{-1}(y)} p(\mathbf{x} = x)$$

where $f^{-1}(y)$ is the set of all elements such that $f(x) = y$. For an invertible function this simplifies to

$$p(\mathbf{y} = y) = p(\mathbf{x} = f^{-1}(y)) \tag{2}$$

(2) is the discrete analogue to (1), but without the log-determinant-Jacobian. Intuitively, the log-det-jacobian corrects for changes of volume in continuous space. Volume is 'accounted for' in the pre-image due to the discrete nature of the sum. (2) is very computationally attractive for many reasons due to this.

Discrete Flow Transformations: Useful discrete invertible transformations must be developed to utilize (2). An example of this is the XOR function. Consider the binary autoregressive case:

$$\mathbf{y}_d = \mu_d \oplus \mathbf{x}_d$$

where μ_d is a function of previous outputs, $\oplus$ is the XOR function. The inverse is $\mathbf{x}_d = \mu_d \oplus \mathbf{y}_d$.

To extend XOR to the categorical setting, consider a D-dimensional vector $\mathbf{x}$, each element of which takes values $0, 1, ..., K-1$. The categorical XOR analogue is:

$$\mathbf{y}_d = (\mu_d + \sigma_d \cdot \mathbf{x}_d) \mod K \tag{3}$$

μ_d and σ_d are autoregresive functions of $\mathbf{y}$ taking values in $0, 1, ..., K-1$ and $1, ..., K-1$ respectively. For this function to be invertible σ and K must be coprime. In most experiments, σ was merely set to 1. At $K = 2, \sigma = 1$, this corresponds to XOR.

Backprop and Optimization: With discrete flow models, the maximum likelihood objective per datapoint is

$$\log p(\mathbf{y}) = \log p(f^{-1}(\mathbf{y}))$$

where the flow f has parameters according to it's autoregressive or bipartite network, and the base distribution p has parameters as a factorized (or autoregressive) distribution. Gradient descent w.r.t. the base distribution is "straightforward". To preform gradient descent with respect to the flow parameters, one must backpropogate through the discrete-output function μ and σ. The authors use the straight-through gradient estimator.

On the forward pass, the authors use both a one-hot and an argmax function, that they backprop back through via replacing it with the Gumbel-sofmax function.

Empirical Results: Nothing to scoff at. While in some experiments, the performance was lower, it was up to 1341x as fast as other methods (Penn Tree Bank). The main benefit of this method seems to be speed with performance near that of continuous flows.

Comments: Given the nature of discrete probability distributions, for any discrete random variable $\mathbf{x}$ it is possible to calculate the MLE in closed-form. However, it is not always possible to factor this distribution, in addition to scaling problems as the number of random variables / observable states increase. Using discrete normalizing flows allows one to approximate or exactly factor distributions more easily. Fully factored distributions are very attractive for directed graphical models.

I neglected bipartite flows on this review and mostly focused on the autoregressive version for simplicity.