20.10.2 Differentiable Generator Nets

Many generative models are based on the idea of using a differentiable **generator network**. The model transforms samples of latent variables $\mathbf{z}$ to samples $\mathbf{x}$ or to distributions over samples $\mathbf{x}$ using a differentiable function $g(\boldsymbol{z}; \boldsymbol{\theta}^{(g)})$ which is typically represented by a neural network. This model class includes variational autoencoders, which pair the generator net with an inference net, generative adversarial networks, which pair the generator network with a discriminator network, and techniques that train generator networks in isolation.

Generator networks are essentially just parametrized computational procedures for generating samples, where the architecture provides the family of possible distributions to sample from and the parameters select a distribution from within that family.

As an example, the standard procedure for drawing samples from a normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ is to feed samples $\boldsymbol{z}$ from a normal distribution with zero mean and identity covariance into a very simple generator network. This generator network contains just one affine layer:

$$\boldsymbol{x} = g(\boldsymbol{z}) = \mu + \boldsymbol{L}\boldsymbol{z} \tag{20.71}$$

where $\boldsymbol{L}$ is given by the Cholesky decomposition of $\boldsymbol{\Sigma}$.

Pseudorandom number generators can also use nonlinear transformations of simple distributions. For example, **inverse transform sampling** (Devroye, 2013) draws a scalar z from $U(0, 1)$ and applies a nonlinear transformation to a scalar x. In this case $g(z)$ is given by the inverse of the cumulative distribution function $F(x) = \int_{-\infty}^{x} p(v)dv$. If we are able to specify $p(x)$, integrate over x, and invert the resulting function, we can sample from $p(x)$ without using machine learning.

To generate samples from more complicated distributions that are difficult to specify directly, difficult to integrate over, or whose resulting integrals are difficult to invert, we use a feedforward network to represent a parametric family of nonlinear functions g, and use training data to infer the parameters selecting the desired function.

We can think of g as providing a nonlinear change of variables that transforms the distribution over $\mathbf{z}$ into the desired distribution over $\mathbf{x}$.

Recall from equation 3.47 that, for invertible, differentiable, continuous g,

$$p_z(\boldsymbol{z}) = p_x(g(\boldsymbol{z})) \left| \det(\frac{\partial g}{\partial \boldsymbol{z}}) \right|. \tag{20.72}$$