Generative Modeling

John Thickstun

Generative modeling ask the following question: how can we (approximately) sample from an unknown probability distribution \( p \) over a space \( \mathcal{X} \), given observations \( x_1, \ldots, x_n \sim p \)? An answer to this question consists of procedure that, when followed, produces samples \( \hat{x} \sim \hat{p} \) where \( \hat{p} \) approximately equals \( p \). A deterministic procedure cannot produce randomness so our procedure will require a source of randomness as input. We won’t worry about (pseudo-)random number generation in this course, and throughout we’ll assume that we have access to samples from the simple distributions Uniform(0, 1) and \( \mathcal{N}(0, 1) \). Formally, an answer to the generative modeling question consists of a function (a generator) \( g : \mathcal{Z} \to \mathcal{X} \) that maps a source of simple randomness \( z \sim q \) to outputs \( \hat{x} = g(z) \sim \hat{p} \) such that \( \hat{p} \approx p \).

Generative modeling is a statistical question, because we only observe \( p \) through samples \( x_i \sim p \).

As a warm-up, we can ask the same question in a simpler setting where \( p \) is known. In this setting, generative modeling becomes a probabilistic question: how can we draw samples from the (known) distribution \( p \) given samples from a simple distribution \( q \)? For example, suppose we want to sample from a Gaussian distribution \( p(x) = \mathcal{N}(x; \mu, \sigma^2) \), given access to samples from \( q(z) = \mathcal{N}(z; 0, 1) \). If we define \( g : \mathcal{Z} \to \mathcal{X} \) by \( z \mapsto \mu + \sigma z \), then \( g(z) \sim \mathcal{N}(\mu, \sigma^2) \). This answers the generative modeling question: given a sample \( z \sim \mathcal{N}(0, 1) \), I can evaluate \( g(z) \) to get a sample from \( \mathcal{N}(\mu, \sigma^2) \). A useful perspective on this process is that the function \( g \) “pushes forward” the distribution \( \mathcal{N}(0, 1) \) on the space \( \mathcal{Z} \) to the distribution \( \mathcal{N}(\mu, \sigma^2) \) on \( \mathcal{X} \).

Pushforward Distributions

The pushforward construction that we saw for normal distributions is a common construction in numerical computing. Low-level software provides a single, simple source of randomness; for our purposes we can assume that this primitive source of randomness is Uniform(0, 1). To generate randomness from other distributions, software relies on pushforward distributions.

**Definition 1.** Given a probability space \((\mathcal{Z}, q)\), a (measurable) function \( g : \mathcal{Z} \to \mathcal{X} \) induces a pushforward distribution on \( \mathcal{X} \) defined, for any (measurable) set \( A \subset \mathcal{X} \) by

\[
\Pr(A) = \int_{g^{-1}(A)} q(z) \, dz.
\]

Suppose we want to sample from a biased coin, i.e. \( x \sim \text{Bernoulli}(p) \). Defining \( g(z) = 1_{z<p} \) with \( z \sim \text{Uniform}(0, 1) \) induces the desired pushforward distribution \( g(z) \sim \text{Bernoulli}(p) \). A more interesting example is the Box-Muller transform [Box and Muller, 1958], which induces a \( \mathcal{N}(0, 1) \) pushforward when applied to inputs \( z \sim \text{Uniform}(0, 1) \). Generating samples \( x \sim p \) using a pushforward of some generic source of randomness is sometimes called simulation of \( p \) in the statistics community. In the machine learning community, the function \( g : \mathcal{Z} \to \mathcal{X} \) is often called a generator.
**Example 1.** If we know the CDF of a distribution $p$ is given by the function $F : \mathbb{R} \to [0, 1]$, then we can specify a pushforward distribution to sample from $p$. Define the inverse CDF by

$$g(z) = \inf \{ x : F(x) \geq z \}. \quad (1)$$

Given $z \sim \text{Uniform}(0, 1)$, it follows that $g(z) \sim p$:

$$\Pr(g(z) \leq x) = \Pr(z \leq F(x)) = F(x). \quad (2)$$

This technique is called inverse transform sampling, or the quantile method, and is based on the observation that the pushforward of the uniform distribution through the inverse-CDF of $p$ is the distribution $p$.

**Example 2.** If $p$ is a discrete distribution on a finite space $\mathcal{X}$, then we can apply inverse transform to sample from $p$ by fixing a (possibly arbitrary) ordering on the elements of $\mathcal{X}$ and constructing a step-wise “CDF,” extending the Bernoulli example that we saw before. Alternatively, we can use the Gumbel-Max generator [Gumbel, 1954] defined by

$$g(z) = \arg \max_x \left( \log p(x) - \log \log \frac{1}{z} \right).$$

If $z \sim \text{Uniform}(0, 1)$ then $g(z) \sim p$.

**Finite Modeling**

Previously, we saw two ways to sample from a known categorical distribution $p(x)$ by constructing a generator that induces $p$ as a pushforward distribution. Now we can begin to address the more interesting question: how do we sample from an unknown distribution $p(x)$ after observing samples $x_1, \ldots, x_n \sim p$ (i.i.d.)? The most direct way to do this in the finite setting is to estimate a probability $\pi_x$ of each element $x \in \mathcal{X}$. To be consistent, we should require that $\sum_x \pi_x = 1$, in which case the estimator $\hat{p}$ defined by $\hat{p}(x) = \pi_x$ is a probability distribution; we can sample from $\hat{p}(x)$ by formulaically constructing the appropriate generator.

What constitute a good estimate of $\pi_x$? A natural choice is the (normalized) empirical count of $x$ in the observed data $x_1, \ldots, x_n$; i.e.

$$\pi_x = \frac{1}{n} \sum_{i=1}^n 1_{x_i = x}.$$

This intuitive estimator is an example of a maximum likelihood estimator (MLE); among all possible categorical distributions in the simplex over $|\mathcal{X}|$ elements, i.e. $\Delta^{|\mathcal{X}|-1}$, $\hat{p}$ is the most likely explanation for the observed data. Formally,

$$\hat{p}(x) = \arg \max_{r \in \Delta^{|\mathcal{X}|-1}} r(x_1, \ldots, x_n) = \arg \max_{r \in \Delta^{|\mathcal{X}|-1}} \frac{1}{n} \sum_{i=1}^n \log r(x_i).$$

This estimator enjoys many desirable properties, e.g. consistency: $\lim_{n \to \infty} \pi_x \to p(x)$ for all $x$.

But there are reasons to be suspicious of the MLE. For example, suppose an element $x$ occurs zero times in our observations $x_1, \ldots, x_n$. The MLE assigns zero probability mass to $x$. This could
be quite undesirable! Quantitatively, if we measure the quality of our estimator \( \hat{p} \) by the KL-divergence \( D(\hat{p} \parallel p) \), then this quantity is infinity if \( \hat{p}(x) = 0 \) and \( p(x) > 0 \). This motivates interest in regularization, e.g. “smoothing” estimators that hedge their bets by allocating at least a small amount of mass to every item in \( \mathcal{X} \) [Chen and Goodman, 1999]. This missing mass problem—where some elements of the distribution are never even observed—is particularly pernicious in domains like NLP, where vocabularies (i.e. \(|\mathcal{X}|\)) are large and the distribution over words (elements of \( \mathcal{X} \)) follow a power law: no matter how large \( n \) is, you are likely to miss something. For an introduction to missing mass estimation, see McAllester and Schapire [2000], and for a modern perspective on smoothing estimators, see Orlitsky and Suresh [2015].

Despite these caveats, the MLE is generally the first estimator we consider. Some of its most extreme failings can be muted by the implicit regularizing effects of a model. But the question of the right estimation objective for generative modeling is something that we will periodically revisit during this course.

**Continuous Modeling**

In contrast to finite modeling, we cannot tabulate the probability (density) at each point in a continuous space. The two traditional methods for simulating continuous distributions are parametric and non-parametric density estimation. The idea is to search for a density among a finite-dimensional (parametric) or infinite-dimensional (non-parametric) family of densities that best fits the observed data (according to, e.g., the MLE criterion). Given a density estimate \( \hat{p}(x) \), we can formulaically construct a generator to sample from \( \hat{p}(x) \) to sample from this distribution.

For an introduction to non-parametric density estimation, and in particular kernel density estimation, see the first few sections of [Tsybakov, 2008]. As a simple example of parametric estimation, suppose we observe samples \( x_1, \ldots, x_n \sim \mathcal{N}(\mu, \sigma^2) \) where the parameters \( \mu \) and \( \sigma^2 \) are unknown. The maximum likelihood estimator of this data is given by \( \mathcal{N}(\hat{\mu}, \hat{\sigma}^2) \) where \( \hat{\mu} \) is the sample mean and \( \hat{\sigma}^2 \) is the sample variance. And we can generate samples from this distribution using the generator \( g(z) = \hat{\mu} + \hat{\sigma}z \), where \( z \sim \mathcal{N}(0, 1) \).

Many of the methods we consider later in this course go beyond the traditional density estimation framework. Recall that our goal is to sample from \( x \sim \hat{p} \), whereas density estimators provide us values \( \hat{p}(x) \). It is usually a mechanical exercise to construct a generator that simulates samples from a given density \( \hat{p}(x) \). But if all we want is sample, construction of a density estimator that allows us to infer \( \hat{p}(x) \) may not be necessary. Later in the course, we will regularly see generative models that cannot do inference: these models allow us to sample \( x \sim \hat{p} \), but the distribution \( \hat{p} \) itself is implicit, and inferring \( \hat{p}(x) \) for a particular value of \( x \) may be quite difficult.

To see why this could be, suppose \( \hat{p} \) is defined implicitly as the pushforward of a density \( q \) by a generator \( g : Z \rightarrow \mathcal{X} \). Changing variables from \( z \) to \( x \), we see that

\[
Pr(A) = Pr(g^{-1}(A)) = \int_{g^{-1}(A)} q(z) \, dz = \int_A q(g^{-1}(x))|\nabla_x g^{-1}(x)| \, dx. \tag{3}
\]

Therefore, the density \( q(z) \) pushes forward to

\[
\hat{p}(x) = q(g^{-1}(x))|\nabla_x g^{-1}(x)|. \tag{4}
\]

For simple generators \( g \), when the inverse and Jacobian are easily computed, we can use Equation (4) to convert a generator into a density estimator. But we are interested in learning rich distributions
over highly structured data. A generator $g$ that accurately models this distribution may not have an easily computable inverse or Jacobian. Nevertheless, the pushforward $\hat{p}$ induced by $g$ could be a very good estimate of $p$ and it is easy to sample by sampling $z \sim q$ and evaluating $g(z) \sim \hat{p}$.

References


