DS 102 Data, Inference, and Decisions **Fall 2019**

Lecture 10: Approximate Inference via Sampling

Lecturer: Ramesh Sridharan

1 Review and Motivation

In the last few lectures, we've considered a situation in which we observe data y that is generated with respect to some hidden or unknown unknown random variables x. We've seen how to use the likelihood function $\mathbb{P}(y|x)$ to update our prior belief $\mathbb{P}(x)$, and obtain a posterior probability $\mathbb{P}(x|y)$ distribution on x after observing the data y:

$$\mathbb{P}(x|y) = \frac{\mathbb{P}(y \,|\, x) \cdot \mathbb{P}(x)}{\mathbb{P}(y)}$$

where $\mathbb{P}(y) = \sum_{x \in \mathcal{X}} \mathbb{P}(y \,|\, x) \cdot \mathbb{P}(x)$ or $\mathbb{P}(y) = \int_{x \in \mathcal{X}} \mathbb{P}(y \,|x) \cdot \mathbb{P}(x) dx$ depending on the context. Previous lectures showed us that if the posterior is from a distribution we know, we can instantiate the normalizing constant later. In general though, when the prior is not a conjugate prior for the likelihood function, calculating $\mathbb{P}(y)$ exactly (and calculating $\mathbb{P}(x|y)$) can be quite difficult. In particular, we might need to compute a sum over an exponential number of possibilities, or an integral with no closed form solution.

The example of posterior density calculations from previous lectures is one of many in which we'd like to be able to calculate, or even sample from, an arbitrary distribution, which may not have a nice distributional form that we're used to. This lecture will detail methods for *approximating* distributions via sampling. In this note, we denote the distributions we want to sample as $\mathbb{P}(x)$, though keep in mind that this distribution could be any distribution, in particular it makes sense to think of it as the posterior density $\mathbb{P}(x|y)$ (it is *not* necessarily a prior distribution on anything). We'll discuss rejection, importance, and Markov Chain Monte Carlo (MCMC) sampling.

2 Rejection Sampling

Imagine you want to sample uniformly from the joint distribution $p(x_1, x_2) \propto \mathbb{I}\{x_1^2 + x_2^2 \leq 1\}$, that is, you want to sample uniformly in area from points (x_1, x_2) that lay within the unit circle (blue region in Figure 10.1). Sampling directly from that distribution is quite hard, and you might wish that you had been asked instead to sample from the unit square (grey region in Figure 10.1) – if this were the task, you could first sample $x_1 \sim Uniform(-1, 1)$ and then sample $x2 \sim Uniform(-1, 1)$ as independent draws, and the concatenation (x_1, x_2) would be drawn uniformly from the unit square.

In fact, we can use the ease of sampling on the unit square to our advantage. In particular, we can sample candidate pairs (z_1, z_2) uniformly from the unit square as above, but then only put them

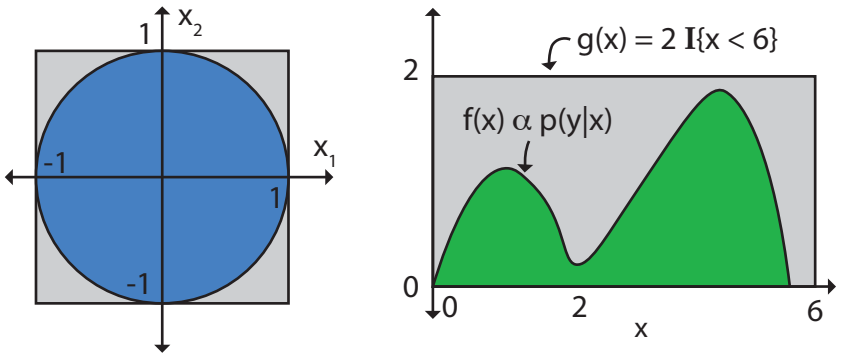

Figure 10.1: Two illustrations of sampling problems. The colored regions denote distributions we'd like to sample from, grey regions denote areas it is easy to sample from.

in our sample if $z_1^2 + z_2^2 \leq 1$. This procedure will only give us points within the unit circle, and furthermore, the probability is uniform across the unit circle.

The key idea with rejection sampling is to generate many samples according to a distribution that is easy to sample from, and then discard samples at a rate to match the second distribution. In the unit circle example, that rate is an indicator function (0 if candidate point (z_1, z_2) falls outside the circle, 1 otherwise). In general, rejection sampling allows a selection rate that is any number in the continuous range $[0, 1]$.

More generally, **rejection sampling** is an algorithm for sampling from a distribution $\mathbb{P}(x) \propto f(x)$, where $f(x)$ is a known function that doesn't have to be a probability distribution (i.e. it doesn't need to integrate to 1). Suppose we also have a function $g(x)$, such that $g(x) \geq f(x), \forall f(x) > 0$. Algorithmically, rejection sampling takes the following steps:

rejection sampling

---

1. repeat:
   (a) Generate sample x from $g(\cdot)$.
   (b) Calculate the acceptance probability: $\alpha = \frac{f(x)}{g(x)}$
   (c) With probability α, put x in the sample, otherwise reject it.

In our unit circle example, $f(x) = \mathbb{I}\{x_2^2 + y_1^2\} \leq 1$ (indicator of being within the unit circle), and $g(x) = \mathbb{I}\{\max(|x_1|, |x_2|) < 1\}$ (indicator of being within the square). In this example, $alpha$ is either 1 or 0, but in general this need not be the case.

In particular, consider a second example where we now want to sample from a univariate posterior probability $\mathbb{P}(x|y) =\propto \mathbb{P}(y|x)\mathbb{P}(x)$, where $\mathbb{P}(x)$ is not a conjugate prior to the likelihood $\mathbb{P}(y|x)$. The posterior probability itself, is unknown, but since we've already specified a prior and a likelihood function, we do know $f(x) = \mathbb{P}(y|x)\mathbb{P}(x)$. We can then use the rejection sampling algorithm above to get a sample from which we can approximate the posterior, as shown in Figure 10.1.

In Figure 10.1, the fraction of time we accept a candidate point drawn from the grey distributions is equal to the area of the colored distributions over the area of the grey distributions. In the example with the posterior density on the right hand side of the figure, that means we reject roughly half of the candidate samples. This might not seem to bad in one dimension, but in high dimensional sampling settings, rejection sampling can be quite wasteful.

3 Importance Sampling

Taking a closer look at Figure 10.1 (right), we see that most of the samples we get from the grey distribution defined by $g(x)$ will actually tell us something about the distribution $f(x)$. Some points (e.g. in $x = 2$) are less likely, so we'd somehow want to down-weight those samples according to their probability of being sampled from a density proportional to $f()$. This is exactly what importance sampling does.

Importance sampling approximates distribution $\mathbb{P}(x)$ by sampling points from a second distribution $\mathbb{Q}(x)$, and weighting them by their likelihood ratio $\mathbb{P}(x)/\mathbb{Q}(x)$:

importance sampling

1. repeat:
   (a) Generate sample $x^{(i)}$ from $g(\cdot)$.
   (b) Calculate the importance weight $w(x^{(i)}) = \frac{\mathbb{P}(x^{(i)})}{\mathbb{Q}(x^{(i)})}$
   (c) Put $x^{(i)}$ in the sample, and store weight $w(x^{(i)})$ as well.

Importance sampling looks very similar to rejection sampling, except now every point we sample from $g(\cdot)$ is getting used in our sample. Then, to estimate say a function $\phi(x)$ in expectation over the population, we would weight each example in our sample average:

$$\mathbb{E}_{x\sim\mathbb{P}(x)}\left[\phi(x)\right] \approx \frac{1}{n}\sum_{i=1}^{n} w(x^{(i)})\phi(x^{(i)})$$

In fact, such an estimate is unbiased. Let $c\cdot\mathbb{Q}(x) = g(x)$ so that $\mathbb{Q}(\cdot)$ denotes the probability density proportional to $g(\cdot)$. Then

$$\begin{aligned}
\mathbb{E}_{x^{(0)},\ldots,x^{(i)}\overset{i.i.d.}{\sim}\mathbb{Q}(x)}\left[\frac{1}{n}\sum_{i=1}^{n} w^{(i)}\phi(x^{(i)})\right] &= \mathbb{E}x^{(0)},\ldots,x^{(i)}\overset{i.i.d.}{\sim}\mathbb{Q}(x)\left[\frac{\mathbb{P}(x)}{\mathbb{Q}(x)}\cdot\phi(x)\right] \\
&= \int_{x\in\mathcal{X}}\frac{\mathbb{P}(x)}{\mathbb{Q}(x)}\phi(x)\mathbb{Q}(x)dx \\
&= \int_{x\in\mathcal{X}}\phi(x)\mathbb{P}(x)dx \\
&= \mathbb{E}_{x\sim\mathbb{P}(x)}\left[\phi(x)\right]
\end{aligned}$$

In practice we'd almost always prefer importance sampling to rejection sampling, as it has lower variance. Importance sampling is especially helpful for sampling low probability events, where

the probability of rejection in rejection samplings would be high. In general, importance sampling ideas are very powerful and show up in many areas of statistics and probability, as well as across scientific domains.

4 Markov Chain Monte Carlo (MCMC)

Rejection sampling and looked at fresh draws of samples x_i. Markov Chain Monte Carlo approaches take a sequential approach to sampling, where previous samples impact your current sample.

The general algorithm for MCMC takes to following iterative form:

1. generate some seed sample x_0, and set $x = x_0$.
2. repeat:
   (a) use previous sample x to generate a new sample x'

The 'Monte Carlo' part of MCMC comes from the repeated sampling aspect; the 'Markov Chain' is because the samples form a Markov Chain with steady-state probabilities $\mathbb{P}(x)$.

We'll define two specific MCMC algorithms: Gibb's sampling and Metropolis-Hastings.

4.1 Metropolis-Hastings

Metropolis-Hastings uses a proposal distribution $\nu(x'|x)$ for generating the next candidate draw from the most recent one. We'll again use the function $f(x)$ which $\mathbb{P}(x)$ is proportional to. The algorithm is as follows (to generate a total of K samples):

Metropolis-Hastings

1. For $t = 1, 2, \ldots, T_{\text{burn-in}}$:
   (a) Generate a proposal x' from proposal distribution $\nu(x'|x)$.
   (b) Calculate the importance weight $\alpha = \min\left(1, \frac{f(x')\cdot\nu(x \mid x')}{f(x)\cdot\nu(x' \mid x)}\right)$
   (c) With probability α, set $x \leftarrow x'$.
2. For $t = T_{\text{burn-in}} + 1, T_{\text{burn-in}} + 2, ..., T_{\text{burn-in}} + K \cdot T_{\text{sample-freq}}$:
   (a) Generate a proposal x' from proposal distribution $\nu(x'|x)$.
   (b) Calculate the importance weight $\alpha = \min\left(1, \frac{f(x')\cdot\nu(x \mid x')}{f(x)\cdot(x' \mid x)}\right)$
   (c) With probability α, set $x \leftarrow x'$.
   (d) If $t - T_{\text{burn-in}}$ is a multiple of $T_{\text{sample-freq}}$, append x to the sample.

Note that steps (c) in the algorithm above look like a rejection-sampling step, with acceptance probability α. Let's take a closer look at the form for α:

$$\alpha(x, x') = \min\left(1, \frac{f(x') \cdot \nu(x \mid x')}{f(x) \cdot \nu(x' \mid x)}\right)$$

The acceptance probability is higher for candidates x' which are more likely under $\mathbb{P}(\cdot) \propto f(\cdot)$. We down-weight candidates that we were very likeley to see by dividing by $\nu(x' \mid x)$. While these are good properties, we exact form for α is chosen so that the steady state probabilities of the resulting Markov Chain is exactly $\mathbb{P}(x)$. To show this, we need to show that $\mathbb{P}(x)$ preserves detail balance of the chain:

$$\begin{aligned}
\mathbb{P}((\text{transition } x \to x' | \text{we're at } x) \cdot \mathbb{P}(x) &= \mathbb{P}(x) \cdot \nu(x' \mid x) \cdot \alpha(x, x')) \\
&= \mathbb{P}(x)\nu(x' \mid x) \min\left(1, \frac{f(x') \cdot \nu(x \mid x')}{f(x) \cdot \nu(x' \mid x)}\right) \\
&= \min\left(\nu(x' \mid x)\mathbb{P}(x), \mathbb{P}(x') \cdot \nu(x \mid x')\right) \\
&= \mathbb{P}(x') \cdot \nu(x \mid x') \min\left(\frac{\nu(x' \mid x)\mathbb{P}(x)}{\mathbb{P}(x') \cdot \nu(x \mid x')}, 1\right) \\
&= \mathbb{P}(x') \cdot \nu(x \mid x') \cdot \alpha(x', x)) \\
&= \mathbb{P}((\text{transition } x \to x' | \text{we're at } x) \cdot \mathbb{P}(x)
\end{aligned}$$

Why the need for the burn in samples and even afterwards, sampling only once every $T_{\text{sample-freq}}$ draws? Remember that in general, it can take many iterations before Markov chains mix. It might seem that the burn-in rate and and the sample frequency are going to disastrously increase sampling time. In practice, this is generally not the case, if we can pick a good proposal distribution $\nu(x'|X)$ from which to sample points along the random walk. In general we'll pick a simple proposal distribution which is easy to sample from, and for which the density is easy and fast to compute.

4.2 Gibb's sampling

Like Metropolis-Hastings, Gibb's sampling is a form of MCMC. IN fact, Gibb's sampling is a speacial case of the Metropolis Hastings algorithm. Gibb's is used for sampling from multivariate distributions, and does so by sampling one coordinate of the multidimensional random variable at a time.

Suppose we want to sample from a joint distribution on n-random variables: $\mathbb{P}(x_1, x_2, ..., x_n \mid y)$. In general, we don't want to sum (or integrate) over all possible x vectors, but we probably are willing to sample from one element x_i of the vector x at a time. The general flow of Gibb's sampling is to fix $n - 1$ of the random variables, and sample the remaining one given according to the joint distribution with these fixed random variables plugged in.

The algorithm proceeds as follows:

Gibb's Sampling

1. Specify an initial sample $x = (x_1, x_2, \ldots, x_n)$.
2. For $t = 1, 2, \ldots, T_{\text{burn-in}}$:
   (a) For $i = 1, \ldots d$:
      - Sample x_i' from $\mathbb{P}(x_i | x_1, ...x_{i-1}, x_{i+1}, \ldots x_n)$.
      - Set $x_i \leftarrow x_i'$
3. For $t = T_{\text{burn-in}} + 1, T_{\text{burn-in}} + 2, ..., T_{\text{burn-in}} + K \cdot T_{\text{sample-freq}}$:
   (a) For $i = 1, \ldots d$:
      - Sample x_i' from $\mathbb{P}(x_i | x_1, ...x_{i-1}, x_{i+1}, \ldots x_n)$.
      - Set $x_i \leftarrow x_i'$
   (b) If $t - T_{\text{burn-in}}$ is a multiple of $T_{\text{sample-freq}}$, append x to the sample.

As an example of where Gibb's sampling is particularly effective, think back to the Gaussian mixture model (GMM) from previous lectures, where the total probability of a sample is given by the prior on the parameters $\mathbb{P}(\mu_1, \mu_2, \pi_1, \pi_2)$, and well as the observed values y and the hidden assignments x. The total probability of any sample is given by: $\mathbb{P}(\mu)\mathbb{P}(x)p(y \mid x, \mu)$. Sampling from this entire distribution seems like a lot of work, but we've seen in previous lectures and discussions that any single parameter or distribution can be solved for in this model, for example $\mathbb{P}(\mu_1 \mid \mu_2, x_1, \ldots, x_n, y_1, \ldots, y_n)$. That is, it's often it's easier to sample one variable at a time.

Gibb's sampling is incredibly widely used because it is so simple. This simplicity comes about because generally if we look at the distribution of one variable at a time, keeping all other variables fixed, this univariate density is much easier to sample from than the general multivariate distribution on n-dimensional variable x.