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

Lecture 11: Approximate Inference via Sampling II

Lecturer: Jacob Steinhardt

11.1 Review and Motivation

Last lecture, we covered two different sampling methods. First, we saw rejection sampling, which works by sampling from one distribution q that envelopes the target distribution p (i.e. $p(x) \leq C \cdot q(x)$ for some constant C), and then accepting or rejecting those samples with probability based on the ratio $\frac{p(x)}{C \cdot q(x)}$. Next, we discussed a related algorithm called importance sampling, which accepts every sample x, but assigns each sample a weight $\frac{p(x)}{q(x)}$.

Today, we'll discuss a family of methods called Markov Chain Monte Carlo (MCMC) algorithms. As we shall see, instead of trying to sample directly from the target distribution p, MCMC methods set up a Markov chain that converges to p.

11.2 Markov Chains

We begin by reviewing the idea of a Markov chain.

A sequence $x_1, \ldots, x_T$ where the distribution of x_t depends only on x_{t-1} is called a **Markov chain**. A Markov chain is always defined by a transition distribution $A(x^{\text{new}}|x^{\text{old}})$, together with an initial state x_1. (Note that, in general, there are situations where the transition distribution might also depend on the current time step t, but for the purposes of this course that will never be the case.)

Example 11.1. We saw one example of a Markov chain in Example 9.1, where we used an Hidden Markov Model (HMM) to model a population of fish in a pond evolving over time. We posited that the only thing that affect the population of fish at time t is the population at time $t-1$ and randomness induced by birth and death processes.

Example 11.2. A random walk is one example of a Markov chain that is used, for instance, in Google's PageRank algorithm. Starting with some webpage x_1, we randomly follow a link on that page. Thus, the transition distribution is defined by picking a random link on the current webpage to get to a new webpage. x_t is the current webpage afer t steps of (randomly) following links. In this case, the Markov chain gives us a measure of importance for each webpage, since we can look at how frequently each webpage is visited. Intuitively, important webpages should be visited more frequently by this random walk since they will have a lot of incoming links.

Example 11.3. The process defined by $x_1 = 0$, $x_t \mid x_{t-1} \sim N(0.9x_{t-1}, 1)$ is a Markov chain. Here, there is randomness (defined by the Gaussian distribution) which will push x_t away from the origin,

but also a scaling-down (from the factor of 0.9) which will tend to keep x_t from straying too far away.

While Markov chains can be used to model data (as in the above examples), in this lecture we will focus on using them to create algorithms. Our understanding of such algorithms will require the following two key concepts.

The first concept is that of the **stationary distribution**. All "nice enough" Markov chains have the property that if T is large enough, the distribution over x_T is "almost independent" of x_1, and converges to some distribution $\bar{p}(x)$ as $T \to \infty$. $\bar{p}(x)$ is called the stationary distribution. (The technical condition for "nice enough" is that the Markov chain is *ergodic*, and this result is known as the *ergodic theorem*.) One way to think about the stationary distribution is as the proportion of time x_t spends in each state as $T \to \infty$.

The second important concept is that of **mixing time**. The mixing time of a Markov chain captures how long it takes for x_T to be close to the stationary distribution. We will not define this formally in this course, but the following examples provide some intuition.

Example 11.4. Suppose we have a deck of cards that starts in some initial configuration x_1. Our transition distribution is defined by performing a riffle shuffle. Here, the mixing time is the number of times we need to shuffle the deck for the deck to be "almost random." There is actually a paper by probabilists Dave Bayer and Persi Diaconis showing that the answer for a 52 card deck is that we need to perform about 7 shuffles.

Example 11.5. Suppose our Markov chain is defined by a random walk on a complete graph with n vertices (including self-loops). That is, every vertex is connected to itself and every other vertex, and at each step we choose an edge to follow uniformly at random. The stationary distribution will be uniform over the vertices. It turns out that the mixing time in this case is just 1 step, since a single step selects a new vertex uniformly at random.

Example 11.6. Suppose our Markov chain is defined by a random walk on a path of length n. Specifically, when x_{t-1} is an internal node, x_t is either the node to the left or to the right of the current node with equal probability. When x_{t-1} is the leftmost node of the path, x_t is the node to its right with probability 1. Similarly, when x_{t-1} is the path's rightmost node, x_t is the node to its left. Here, the stationary distribution will assign more probability weight on the internal nodes than the edge nodes. The mixing time must be at least n, since the random walk could not possible make it from one end of the path to the other in less than n steps. In fact, it turns out that the mixing time is roughly n^2, since the random walk will tend to go back and forth many times before making it from one end to the other.

11.3 Markov Chain Monte Carlo (MCMC)

Markov Chain Monte Carlo (MCMC) methods are a class of algorithms for sampling from a target probability distribution by constructing a Markov chain. MCMC approaches take a sequential approach to sampling, where previous samples impact your current sample.

In general, MCMC algorithms take the following iterative form:

1. generate some seed sample x_1, and set $x^{\text{old}} = x_1$.
2. repeat:
   (a) use previous sample x^{old} to generate a new sample x^{new}

The 'Monte Carlo' part of MCMC comes from the repeated sampling aspect; the 'Markov Chain' is because the samples form a Markov chain. In general, we aim to choose the transition distribution so that this Markov chain has the target distribution p as its stationary distribution. in the following sections, we will define two specific MCMC algorithms that accomplish this: Gibb's sampling and Metropolis-Hastings.

11.3.1 Gibb's sampling

We begin with a motivating example. Suppose we have an arbitrary distribution $p(x_1, \ldots, x_n)$ from which we want to sample. We could try doing rejection sampling using some proposal distribution $q(x_1, \ldots, x_n)$ for all the x_i at once. However, rejection sampling will tend to be much too slow – typically, the acceptance rate will be exponentially small in n. For example, even if the x_i are independent and $\frac{q(x_i)}{p(x_i)} \leq 1.1$, we need 1.1^n tries (which is approximately $2.4 \cdot 10^4 1$ for $n = 1000$) before making an acceptance. In general, rejection sampling suffers from a "curse of dimensionality." Importance sampling can help address this issue somewhat, but unfortunately we can still run into problems where the majority of the samples are assigned exponentially small weights.

Gibb's sampling is used for sampling from such multivariate distributions, and does so by sampling one coordinate of the multidimensional random variable at a time. When we sample only one x_i at a time, we will no longer by sampling directly from p, but if we keep doing so iteratively, this process will define a Markov chain with the right stationary distribution.

The general flow of Gibb's sampling is to fix all but one of the random variables, and sample the remaining one according to its conditional distribution with everything else fixed. We can outline this algorithm as follows:

- Initialize $(x_1, \ldots, x_n)$ arbitrarily.
  Repeat:
  - Pick i (randomly or sequentially)
  - Re-sample x_i from $p(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$
- Pick i (randomly or

Example 11.7. Hierarchical Bayesian models are an example where Gibb's sampling is often particularly effective. For instance, 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 x and the hidden variables z. The total probability of any sample is given by: $\mathbb{P}(\mu)\mathbb{P}(z)p(x \mid z, \mu)$. 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 closed-form in this model. For example, $\mathbb{P}(\mu_1 \mid \mu_2, z_1, \ldots, z_n, x_1, \ldots, x_n)$. 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, when 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 full multivariate distribution. Moreover, Gibb's sampling is very flexibly; unlike EM, for example, we do not need to "get lucky" with the graphical model structure for the updates to be efficient.

However, one important drawback to MCMC methods, including Gibb's sampling, is that we have to worry about mixing time. Practically, we often to run the chain for a large number of steps (called "burn-in") before we can begin collecting samples. In general, it can take many iterations before Markov chains start to mix. Even then, we often only sample once every $T_{\text{sample-freq}}$ steps in the chain. A more formal version of the Gibb's sampling algorithm which accounts for the burn-in period and sampling frequency is given below.

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 $p(x_i | x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n)$.
   - Set $x_i \leftarrow x_i'$
3. For $t = T_{\text{burn-in}} + 1, T_{\text{burn-in}} + 2, \ldots, T_{\text{burn-in}} + K \cdot T_{\text{sample-freq}}$:
   (a) For $i = 1, \ldots d$:
   - Sample x_i' from $p(x_i | x_1, \ldots x_{i-1}, x_{i+1}, \ldots x_n)$.
   - 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.

How does Gibb's sampling compare to other algorithms we have seen in the course so far, like EM? Since we can never know for certain that we have run our Markov chain long enough to deal with mixing time, EM can be more reliable. However, EM is only guaranteed to converge to a local optimum; we can never be sure that it converges to a *global* optimum. In general, each approach has its own tade-offs, so which approach we choose should always be based on what we know about our specific application, and which concerns (e.g. time until convergence, mixing time, getting stuck in local optima) we are most worried about for our current problem. We will discuss many of these trade-offs in more detail in the context of a real example from computational biology in the next lecture.

11.3.2 Metropolis-Hastings

The Gibb's sampling algorithm was based on just one possible Markov chain that has the desired stationary distribution. Is there a more general strategy? It turns out that, by marrying the ideas of MCMC and rejection sampling, we can combine *any* proposed Markov chain $q(x^{\text{new}}|x^{\text{old}})$ with an accept/reject step to create a new Markov chain with the correct stationary distribution. This approach is known as **Metropolis-Hastings**.

Metropolis-Hastings uses a proposal distribution $q(x^{\text{new}}|x^{\text{old}})$ for generating the next candidate draw from the most recent one. 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^{new} from proposal distribution $q(x^{\text{new}}|x^{\text{old}})$.
   (b) Calculate the importance weight $\alpha = \min\left(1, \frac{p(x^{\text{new}}) \cdot q(x^{\text{old}} \mid x^{\text{new}})}{p(x^{\text{old}}) \cdot q(x^{\text{new}} \mid x^{\text{old}})}\right)$
   (c) With probability α, set $x^{\text{old}} \leftarrow x^{\text{new}}$.
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^{new} from proposal distribution $q(x^{\text{new}}|x^{\text{old}})$.
   (b) Calculate the importance weight $\alpha = \min\left(1, \frac{p(x^{\text{new}}) \cdot q(x^{\text{old}} \mid x^{\text{new}})}{p(x^{\text{old}}) \cdot q(x^{\text{new}} \mid x^{\text{old}})}\right)$
   (c) With probability α, set $x^{\text{old}} \leftarrow x^{\text{new}}$.
   (d) If $t - T_{\text{burn-in}}$ is a multiple of $T_{\text{sample-freq}}$, append x^{old} 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:

$$\alpha(x^{\text{old}}, x^{\text{new}}) = \min\left(1, \frac{p(x^{\text{new}}) \cdot q(x^{\text{old}} \mid x^{\text{new}})}{p(x^{\text{old}}) \cdot q(x^{\text{new}} \mid x^{\text{old}})}\right).$$

The acceptance probability is higher for candidates x^{new} which are more likely under $p(\cdot)$. We also down-weight candidates that were very likely to see according to the proposal distribution q by dividing by $q(x^{\text{new}} \mid x^{\text{old}})$. Finally, taking the minimum with 1 ensures that α is always a valid probability.

While these are nice properties for the acceptance probability, we ultimately choose this exact form for α so that the stationary distribution of the resulting Markov chain is exactly $p(x)$. It is also interesting to note that Gibb's sampling is a special case where q is such that we always accept the proposals.

As in our discussion of Gibb's sampling, we may be concerned about mixing time. It might seem that the burn-in rate and sample frequency will disastrously increase overall sampling time. In practice, this is generally not the case; if we can pick a good proposal distribution $q(x^{\text{new}}|x^{\text{old}})$. 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.