Bayesian Inference: Gibbs Sampling

Ilker Yildirim
Department of Brain and Cognitive Sciences
University of Rochester
Rochester, NY 14627

August 2012

References: Most of the material in this note was taken from: (1) Lynch, S. M. (2007). *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*. New York: Springer; and (2) Taylan Cemgil's lecture slides on Monte Carlo methods (http://www.cmpe.boun.edu.tr/courses/cmpe58n/fall2009/)

1. Introduction

As Bayesian models of cognitive phenomena become more sophisticated, the need for efficient inference methods becomes more urgent. In a nutshell, the goal of Bayesian inference is to maintain a full posterior probability distribution over a set of random variables. However, maintaining and using this distribution often involves computing integrals which, for most non-trivial models, is intractable. Sampling algorithms based on Monte Carlo Markov Chain (MCMC) techniques are one possible way to go about inference in such models.

The underlying logic of MCMC sampling is that we can estimate any desired expectation by ergodic averages. That is, we can compute any statistic of a posterior distribution as long as we have N simulated samples from that distribution:

$$E[f(s)]_{\mathcal{P}} \approx \frac{1}{N}\sum_{i=1}^{N} f(s^{(i)}) \tag{1}$$

where $\mathcal{P}$ is the posterior distribution of interest, $f(s)$ is the desired expectation, and $f(s^{(i)})$ is the i^{th} simulated sample from $\mathcal{P}$. For example, we can estimate the mean by $E[x]_{\mathcal{P}} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$.

How do we obtain samples from the posterior distribution? Gibbs sampling is one MCMC technique suitable for the task. The idea in Gibbs sampling is to generate posterior samples by sweeping through each variable (or block of variables) to sample from its conditional distribution with the remaining variables fixed to their current values. For instance, consider the random variables X_1, X_2, and X_3. We start by setting these variables to their initial values $x_1^{(0)}$, $x_2^{(0)}$, and $x_3^{(0)}$ (often values sampled from a prior distribution q). At iteration i, we sample $x_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)})$, sample $x_2 \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)})$, and sample $x_3 \sim p(X_3 = x_3 | X_1 = x_1^{(i)}, X_2 = x_2^{(i)})$. This process continues until "convergence" (the sample values have the same distribution as if they were sampled from the true posterior joint distribution). Algorithm 1 details a generic Gibbs sampler.

Algorithm 1 Gibbs sampler

Initialize $x^{(0)} \sim q(x)$
for iteration $i = 1, 2, \ldots$ **do**
  $x_1^{(i)} \sim p(X_1 = x_1 | X_2 = x_2^{(i-1)}, X_3 = x_3^{(i-1)}, \ldots, X_D = x_D^{(i-1)})$
  $x_2^{(i)} \sim p(X_2 = x_2 | X_1 = x_1^{(i)}, X_3 = x_3^{(i-1)}, \ldots, X_D = x_D^{(i-1)})$
  $\vdots$
  $x_D^{(i)} \sim p(X_D = x_D | X_1 = x_1^{(i)}, X_2 = x_2^{(i)}, \ldots, X_D = x_{D-1}^{(i)})$
end for

In Algorithm 1, we are not directly sampling from the posterior distribution itself. Rather, we simulate samples by sweeping through all the posterior conditionals, one random variable at a time. Because we initialize the algorithm with random values, the samples simulated based on this algorithm at early iterations may not necessarily be representative of the actual posterior distribution. However, the theory of MCMC guarantees that the stationary distribution of the samples generated under Algorithm 1 is the target joint posterior that we are interested in (Gilks et al., 1996; also see the Computational Cognition Cheat Sheet on Metropolis-Hastings sampling). For this reason, MCMC algorithms are typically run for a large number of iterations (in the hope that convergence to the target posterior will be achieved). Because samples from the early iterations are not from the target posterior, it is common to discard these samples. The discarded iterations are often referred to as the "burn-in" period.

A good way of learning about MCMC techniques is to study examples. In the rest of this note, we develop a Gibbs sampler for a change-point model.

2. A change-point model

Suppose that we observe a sequence of counts $x_1, x_2, \ldots, x_N$ where the average of the counts has some value for time steps 1 to n, and a different value for time steps $n+1$ to N. We model the counts at each time step i as a Poisson variable, which has the following density function:

$$\begin{aligned} \text{Poisson}(x; \lambda) &= e^{-\lambda} \frac{\lambda^x}{x!} \\ &= \exp(x \log \lambda - \lambda - \log(x!)) \end{aligned} \tag{2}$$

where λ is the mean of the distribution. We model the mean λ as a Gamma distribution, which has the following density function:

$$\begin{aligned} \text{Gamma}(\lambda; a, b) &= \frac{1}{\Gamma(a)} b^a \lambda^{a-1} \exp(-b\lambda) \\ &= \exp((a-1) \log \lambda - b\lambda - \log \Gamma(a) + a \log b) \end{aligned} \tag{3}$$

The initial mean λ_1 jumps to a new value λ_2 after a random time step n. Thus the generative model is as follows:

$$\begin{aligned} n &\sim \text{Uniform}(1, 2, \ldots, N) \\ \lambda_i &\sim \text{Gamma}(\lambda_i; a, b) \\ x_i &\sim \begin{cases} \text{Poisson}(x_i; \lambda_1) & 1 \leq i \leq n \\ \text{Poisson}(x_i; \lambda_2) & n < i \leq N \end{cases} \end{aligned} \tag{4}$$

The problem of inferring the posterior distribution over the latent variables n, λ_1, λ_2 can be solved via Bayes theorem.

$$p(\lambda_1, \lambda_2, n | x_{1:N}) \propto p(x_{1:n}|\lambda_1) p(x_{n+1:N}|\lambda_2) p(\lambda_1) p(\lambda_2) p(n) \tag{5}$$

3. Conditional distributions

As Algorithm 1 illustrates, we need the posterior conditionals for each of the variables to perform Gibbs sampling. We start by deriving the full joint distribution. We then derive the posterior conditionals for each of the variables λ_1, λ_2, n using the full joint distribution.

A form of the full joint distribution is given on the right hand side of Equation 5. We start our derivation from there.

$$\begin{aligned} & p(x_{1:n}|\lambda_1) p(x_{n+1:N}|\lambda_2) p(\lambda_1) p(\lambda_2) p(n) \\ &= \left(\prod_{i=1}^{n} p(x_i|\lambda_1) \right) \left(\prod_{i=n+1}^{N} p(x_i|\lambda_2) \right) p(\lambda_1) p(\lambda_2) p(n) \end{aligned} \tag{6}$$

Next we write the log of the full posterior by taking the logarithm of the right hand side of Equation 6 and plugging in Equations 2 and 3 wherever appropriate.

$$\begin{aligned} & \log p(x_{1:n}|\lambda_1) + \log p(x_{n+1:N}|\lambda_2) + \log p(\lambda_1) + \log p(\lambda_2) + \log p(n) \\ &= \sum_{i=1}^{n} (x_i \log \lambda_1 - \lambda_1 - \log(x_i!)) \\ &+ \sum_{i=n+1}^{N} (x_i \log \lambda_2 - \lambda_2 - \log(x_i!)) \\ &+ (a-1) \log \lambda_1 - b\lambda_1 - \log \Gamma(a) + a \log b \\ &+ (a-1) \log \lambda_2 - b\lambda_2 - \log \Gamma(a) + a \log b \\ &- \log N \end{aligned} \tag{7}$$

Now we obtain the posterior conditionals for each of the latent variables by collecting the terms in the full joint distribution that include the latent variable of interest. We start

with λ_1. We obtain its posterior conditional by picking up the relevant terms in Equation 7 (and rearranging some of these terms).

$$\begin{aligned}
&\log p(\lambda_1|n, \lambda_2, x_{1:N}) \\
&=^+ \sum_{i=1}^{n} (x_i \log \lambda_1 - \lambda_1) + (a-1) \log \lambda_1 - b\lambda_1 \\
&= \left(a + \sum_{i=1}^{n} x_i - 1\right) \log \lambda_1 - (n+b)\lambda_1 \\
&=^+ \log \text{Gamma}\left(a + \sum_{i=1}^{n} x_i, n+b\right)
\end{aligned} \tag{8}$$

where the operator $=^+$ means "equal up to a constant". Note that the posterior conditional and the prior for λ_1 are both Gamma distributions (albeit with different sets of parameters). This correspondence of distributions is not random, but arises because we used a "conjugate" prior distribution. That is, for certain pairs of priors and likelihoods, the posterior ends up having the same probability distribution as the prior (with updated parameters). Gamma-Poisson is one such pair of conjugacy, resulting in a Gamma posterior.

The posterior conditional for λ_2 can be derived similarly:

$$\begin{aligned}
&\log p(\lambda_2|n, \lambda_1, x_{1:N}) \\
&=^+ \sum_{i=n+1}^{N} (x_i \log \lambda_2 - \lambda_2) + (a-1) \log \lambda_2 - b\lambda_2 \\
&=^+ \log \text{Gamma}\left(a + \sum_{i=n+1}^{N} x_i, N-n+b\right)
\end{aligned} \tag{9}$$

Finally, we derive the posterior conditional for n, the time step at which counts jump from a mean of λ_1 to a mean of λ_2:

$$\begin{aligned}
&\log p(n|\lambda_1, \lambda_2, x_{1:N}) \\
&= \sum_{i=1}^{n} (x_i \log \lambda_1 - \lambda_1 - \log(x_i!)) + \sum_{i=n+1}^{N} (x_i \log \lambda_2 - \lambda_2 - \log(x_i!)) \\
&=^+ \left(\sum_{i=1}^{n} x_i\right) \log \lambda_1 - n\lambda_1 + \left(\sum_{i=n+1}^{N} x_i\right) \log \lambda_2 - (N-n)\lambda_2
\end{aligned} \tag{10}$$

Note that the conditional posterior for n is not of a known closed form. But we can easily obtain a multinomial distribution for n by computing $p(n|\lambda_1, \lambda_2, x_{1:N})$ for $n = 1, \ldots, N$ which we can use to draw new samples.

Now that we have the posterior conditionals for all the latent variables, we are ready to simulate samples from the target joint posterior in Equation 5 using Algorithm 1.

4. Results

We implemented a Gibbs sampler for the change-point model using the Python programming language. This code can be found on the Computational Cognition Cheat Sheet website.

We start by simulating data from the generative process described in Equation 4 (see Figure 1, top row). Our simulations are based on this synthetic data set.

We initialized our Gibbs sampling chain by sampling each variable from its prior distribution. For n, we drew an integer between 1 and N from Uniform$(1, \ldots, N)$. (In all our simulations we set $N = 50$.) We initialized λ_1 and λ_2 by drawing two independent values from Gamma(a, b). (We set $a = 2$ and $b = 1$ in all our simulations, both for simulating synthetic data set and for the Gibbs sampler.)

We report the results for one MCMC chain, which are typical of other chains that we simulated. We ran the chain for 5200 iterations. We discarded the first 200 iterations as burn-in. Our results are based on the remaining 5000 iterations.

Figure 1 illustrates the results. See its caption for explanation.

5. Summary

Given a generative model for a set of random variables, we can summarize Gibbs sampling in two steps:

- Step 1: Derive the full joint density, and the posterior conditionals for each of the random variables in the model.
- Step 2: Simulate samples from the posterior joint distribution based on the posterior conditionals (Algorithm 1).

We illustrated both of these steps in a change-point detection problem.

Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov Chain Monte Carlo in Practice*. London: Chapman and Hall.

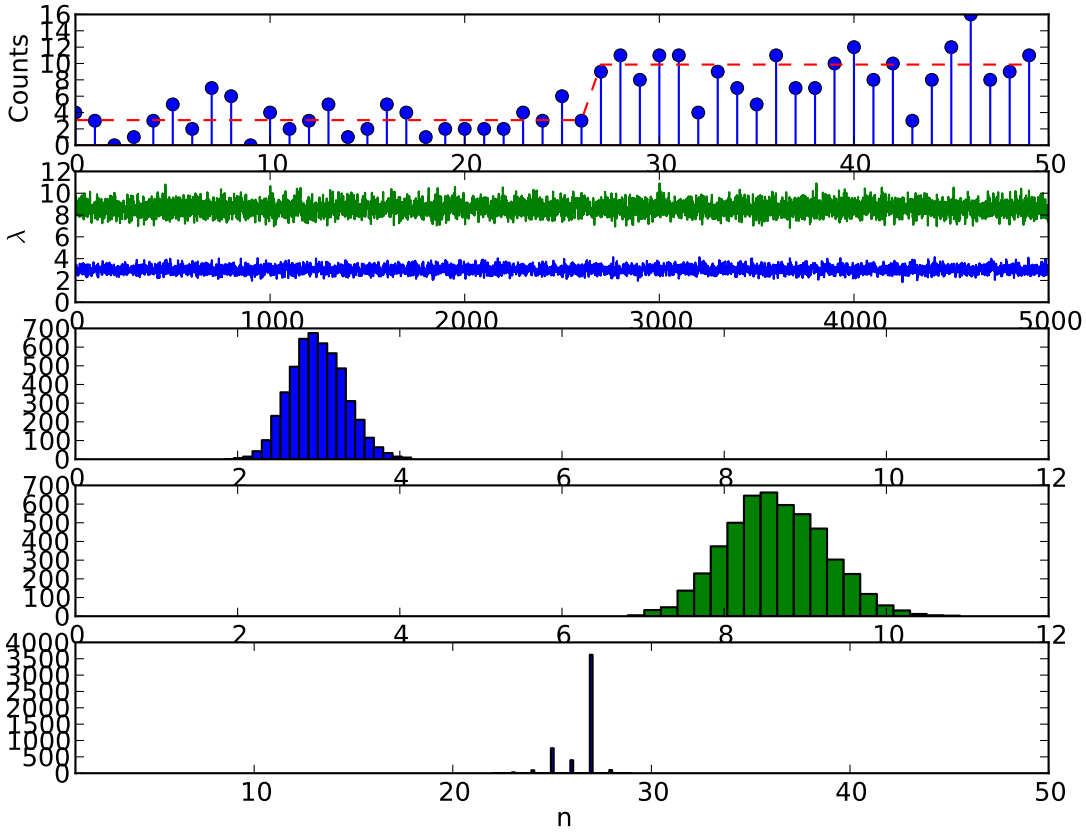

Figure 1: (Top row) The synthetic data set. Each stick is a count, $x_{1:N}$. The dotted red line shows the average count values. This average jumps from λ_1 to λ_2 at $n = 26$. (Row 2) A trace plot based on the posterior samples for λ_1 and λ_2. The Gibbs sampler remarkably recovers the values that actually generated the data set. (Row 3) A histogram plot of the posterior samples for λ_1. (Row 4) A histogram plot of the posterior samples for λ_2. (Bottom row) A histogram plot for n, the change-point. The model is almost certain on the generating value of n.