Basis Expansion Monte Carlo

Eric Kernfeld *

University of Washington, Seattle, WA, USA

November 15, 2014

Abstract

Most Monte Carlo inference methods are left to run until the markov chain reaches its steady state. This leaves the user with a large chain of samples from the distribution of interest. We introduce Basis Expansion Monte Carlo, which runs the sampler piecewise starting at different places in the parameter space in order extract information more quickly. To make inference about the steady state, we gradually update an approximation of the hidden linear operator that underlies any Metropolis-Hastings or Gibbs sampler. We use the steady-state of the approximate operator in place of the true steady-state. Results show ...

1 Introduction

In many statistical models, it is impossible to find a closed form for the distribution of interest (we will call this π). One work-around, originating in computational physics, relies on the fact that for points x_1 and x_2 in the parameter space, $\pi(x_1)/\pi(x_2)$ may still be calculable, though $\pi(x_1)$ and $\pi(x_2)$ are not.

More and references about history, background, and/or tutorials on monte carlo methods

In this setting, we will consider two classes of algorithms. The first, the Metropolis-Hastings scheme, consists of the following procedure.

Algorithm 1: Metropolis-Hastings algorithm

Set $x_0 = 0, i = 0$
Repeat ad nauseum:
  Increment i
  Draw x from a proposal distribution $q(x|x_{i-1})$
  Set $\alpha(x|x_{i-1}) = 1 - min(1, \frac{\pi(x)q(x_{i-1}|x)}{\pi(x_{i-1})q(x|x_{i-1})})$
  Draw u from a uniform density on $[0, 1]$. Set $x_i = x$ with probability $1 - \alpha$, i.e. if $u > \alpha$, and $x_i = x_{i-1}$ otherwise.

Suppose this MCMC algorithm produces a chain $x_1, x_2, x_3, ...$ of samples. Because the algorithm is stochastic, these samples can be viewed as realizations of random variables $X_1, X_2, X_3, ...$ with marginal density functions f_1, f_2, f_3, etc. (Perhaps X_1 is just a constant.) We can write the conditional density of X_2 given X_1 as $f_{2|1}(x_2, x_1) = (1 - \alpha(x_2|x_1))\delta_{x_1}(x_2) + \alpha(x_2|x_1)q(x_2|x_1)$. Then, we have that $f_2(x_2) = \int f_{2|1}(x_2, x_1)f_1(x_1)dx_1$; in more generality, $f_i(x_i) = \int f_{i|i-1}(x_i, x_{i-1})f_{i-1}(x_{i-1})dx_{i-1}$. Noting that $f_{i|i-1}$ doesn't depend on i, we can replace it with a function K so that $f_i(x_i) = \int K(x_i, x_{i-1})f_{i-1}(x_{i-1})dx_{i-1}$. This is a fixed linear operator L that produces f_i as Lf_{i-1}, analogous to the transition probability matrices of discrete-space Markov chain theory. As L cannot be observed directly, we refer to it as the hidden action of an M-H algorithm.

*Electronic address: ekernf01@u.washington.edu; Corresponding author

The object of interest in Bayesian statistics is the steady state of this operator, an eigenfunction π that has eigenvalue 1 so that for any x, $\pi(x) = \int K(x,t)\pi(t)dt$. In BEMC, we approximate L, then compute π from the approximation.

To approximate the hidden action of an M-H sampler on a domain Ω, consider some functions $\{h_i\}_{i=1}^B$ from Ω to $\mathbb{R}$. Suppose they are orthogonal with respect to the L_2 inner product, i.e. $\int h_i(x)h_j(x)dx = 0$ when $i \neq j$. For $\Omega = \mathbb{R}^n$, we might use Gaussian-weighted Hermite polynomials. Consider also a function α from Ω^2 to $[0,1]$ and a matrix M in $\mathbb{R}^{B\times B}$. We will attempt to set things up so that $L \approx \hat{L}_\alpha + \hat{L}_M$, where $(\hat{L}_\alpha f)(x) = \hat{\alpha}(x)f(x)$ and $(\hat{L}_M f)(x) = \sum_{i,j=1}^B h_i(x)M_{ij}\int h_j(x)f(x)dx$. The first term mimics the rejection probability, while the next term tracks movement. At this point in the narration, $\hat{\alpha}$ and M are unknown–strategies to estimate them follow. Even if they were chosen optimally, L may not take the same form as $\hat{L}_\alpha + \hat{L}_M$, so the estimate π may not be correct.

Among other tasks, we need to somehow estimate α, the rejection probability. Fortunately, it is easy to tell when the sampler rejects and when it doesn't. Suppose for a moment that we start the sampler at a point z and it takes a single step to w. If $w \neq z$, then the sampler has shown less of a tendency to reject at z; we can label z with a 1 to estimate α later. If it had stayed at z, we would label it 0. Once the sample space is covered in zeroes and ones, there are many probabilistic classifier methods that could give an estimate of α.

Meanwhile, whenever the sampler moves, we gain information about L_M. To make use of it, notice that the orthogonality of the basis functions implies $\int h_i(x)(L_M h_j)(x)dx = M_{ij}$. This can be written as an expectation $M_{ij} = E_{L_M h_j}[h_i]$, which motivates us to sample from $L_M h_j$ and approximate M_{ij} as a sum. If we can sample from h_j, all we need to do is run each point through the M-H algorithm once, and we will have samples from $L_M h_j$. (If the M-H algorithm rejects, we will not update M_{ij}, because that sample is accounted for by L_α).

How do we sample from h, a basis function that sometimes takes negative values? How do we formally take an expectation? The important property to preserve is the law of large numbers: sample averages of some functions should still converge to their expectation. We use a classic tactic from analysis. Let h_+ be defined as $c_+^{-1}max(h,0)$ and let h_- be defined as $-c_-^{-1}min(h,0)$, with c_+ and c_- chosen so h_+ and h_- each integrate to one. Then define $E_h[f]$ as $c_+E_{h_+}[f] - c_-E_{h_-}[f]$. We can approximate this expectation by sampling z_{n+} from $h_+, n = 1...N_+$ and $z_{n-}, n = 1...N_-$ from h_-. We would then compute $E_h[f] \approx \frac{c_+}{N_+}\sum f(z_{n+}) - \frac{c_-}{N_-}\sum f(z_{n-})$. The optimal allocation of samples between h_+ and h_- minimizes the overall variance, $\frac{c_+^2}{N_+}Var_{h_+}[f] + \frac{c_-^2}{N_-}Var_{h_-}[f]$. To sample from h_+ and h_-, which may not have closed-form inverse CDF's, we employ rejection sampling.

To take care of one last detail, suppose h is $L_M g$ for some g, and we can only sample from h by running an M-H iteration on samples from g. We can still sample from

Algorithm 2: BEMC algorithm

Set M to 0.
Set $T = \{\}$. T will be the training set for α.
For $b_{in} = 1 : B$
  For $b_{out} = 1 : B$
    For $n = 1 : N$
      Draw a sample z_n from $h_{b_{in}}$.
      Run the sampler for one round on z_n. Call the result w_n.
      If $z_n = w_n$:
        Add $(z_n, 0)$ to T.
      Otherwise:
        Add $(z_n, 1)$ to T.
        Increment $M_{b_{out},b_{in}}$ by $h_{b_{in}(w_n)}/N$.

This approximation can also be adapted to Gibbs sampling, a ubiquitous MCMC variant.