Hamiltonian Monte Carlo

Ilan Man and Sanjay Hariharan

April 13, 2016

1 Introduction

Hamiltonian Monte Carlo is a novel approach to Posterior estimation. It improves on the simple random-walk proposal of the Metropolis algorithm, by ensuring that these proposals are in the direction of steepest ascent (reference). In doing so, we avoid the slow exploration of the state space from these random proposals. HMC reduces the correlation between successive sampled states by using Hamiltonian evolution between states, a concept derived from quantum mechanics. The energy preserving aspect of Hamiltonian dynamics will be extremely useful in our derivation and implementation of this algorithm.

2 MCMC review

Markov Chain Monte Carlo methods are a class of algorithms for sampling from a proabability distribution based on constructing a Markov Chain. In Bayesian Inference, we obtain estimates for model parameters by sampling from the posterior distribution. MCMC methods are used extensively in calculating posterior distributions, which are often complex and intractable, and therefore must be approximated numerically. MCMC algorithms have certain attractive properties such as convergence to a limiting distribution, which we exploit in solving Bayesian Inference problems.

2.1 Motivation

In the case of Heirarchical Bayesian Models, we often find the posterior distribution to be comprised of several parameters and is often intractable to solve. Even when tractable, we may find ourselves having to compute several complex integrals which require massive amounts of computing time. In real-world applications, practitioners commonly use MCMC methods to solve for these parameters. There are many flavors of MCMC and the modeler should identify which one to use in which case.

2.2 Example: Gibbs Sampling

Gibbs sampling is a specific variant of MCMC where each parameter of interest can be sampled directly, when conditioned across all the other parameters. This algorithm generates an instance from the distribution of each variable in turn, conditional on the current values of the other variables. This sequence of samples constitutes a Markov Chain whose stationary distribution is the sought-after joint distribution.

2.3 Gibbs Sampling Pseudo-Code

1. set t = 0
2. generate an initial state $x^{(0)} \sim \pi^{(0)}$
3. repeat until t = M
   a. set t = t+1
   b. for each dimension i = 1..D
   c. draw x_i from $p(x_i|x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_D)$

2.4 Shortcomings

The main shortcoming of Gibbs Sampling is the successive autocorrelation of the samples. Due to the fact that the sampling generates a Markov Chain, successive samples are correlated with each other. *Thinning* and *Block Sampling* may help reduce this correlation, but these heuristics sometimes don't work well or require lots of data. Another shortfall specific to Gibbs Sampling is that it requires identifying conditional distributions for each parameter. As with the posterior, some of these conditional distributions may have complex, intractable forms and will need other MCMC methods to compute them.

2.5 Example: Metropolis-Hastings

The Metropolis algorithm is a generalization of Gibbs Sampling, useful when you cannot compute conditional sampling distributions in closed form. At each iteration of this algorithm, we propose a *candidate* for the next sample value of the parameter in question. The most common proposal method is a simple random walk, i.e. we sample from a Standard Normal Distribution. Then, with probability α the candidate is either accepted (updating our chain's next value with the candidate) or rejected (making the chain's next value equal to the current). α is the ratio of the posterior distribution with the proposed parameter over the posterior distribution with the previous parameter, truncated to 1.

2.6 Metropolis-Hastings Pseudo-Code

1. set t = 0
2. generate an initial state $x^{(0)} \sim \pi^{(0)}$
3. repeat until t = M
   a. set t = t+1
   b. generate a proposal state x^* from $q(x|x^{(t-1)})$
   c. calculate the proposal correction factor c = $\frac{q(x^{(t-1)}|x^*)}{q(x^*|x^{(t-1)})}$
   d. calculate the acceptance probability $\alpha = \min\left(1, \frac{p(x^*)}{p(x^{(t-1)})} \times c\right)$
   e. draw a random number u from Unif(0,1)
      i. if u $\leq \alpha$ accept the proposal state x^* and set $x^{(t)} = x^*$
      ii. else set $x^{(t)} = x^{(t-1)}$

2.7 Shortcomings

The main shortcoming of the Metropolis Algorithm is that it explores the posterior space very slowly. This is due to the random walk behavior of the proposals. This makes the algorithm inefficient and requires a lot of computation to accurately estimate low-variance parameters.

3 Hamiltonian Dynamics

Before we can develop Hamiltonian Monte Carlo, we must become familiar with the concept of Hamiltonian dynamics. Hamiltonian dymanics describes an object's motion in terms of its location x and momentum p. Recall from high school physics that momentum p is equal to an object's mass m times its velocity v. For each location an object takes, there is an associated potential energy $U(x)$, and for each momentum there is an associated kineric energy $K(p)$. The total energy of the system is **constant** and is known as the Hamiltonian $H(x, p)$, defined as:

$H(x, p) = U(x) + K(p)$

This description is implemented quantitatively via a set of differential equations known as the Hamiltonian equations:

$\frac{\partial x_i}{\partial t} = \frac{\partial H}{\partial p_i} = \frac{\partial K(\mathbf{p})}{\partial p_i}$
$\frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial x_i} = -\frac{\partial U(\mathbf{x})}{\partial x_i}$

If we can solve for these partial derivatives, we can apply Hamiltonian Dynamics to more general cases. In order to simulate these dynamics numerically for computation, it is necessary to approximate the Hamiltonian equations by discretizing time. This is done by splitting up the interval T into a series of smaller intervals of length δ. The smaller the value of δ the closer the approximation is to the dynamics in continuous time, but the more computationally expensive is the procedure.

3.1 The Leap Frog Method

The Leap Frog Method is a popular method for numerically integrating differential equations, such as the Hamiltonian equations. The method updates the momentum and position variables sequentially, starting by simulating the momentum dynamics over a small interval of time $\frac{\delta}{2}$, then simulating the position dynamics over a slightly longer interval in time δ, then completing the momentum simulation over another small interval of time $\frac{\delta}{2}$ so that x and p now exist at the same point in time. Specifically:

1. Take a half step in time to update the momentum variable:

$p_i(t + \delta/2) = p_i(t) - (\delta/2)\frac{\partial U}{\partial x_i(t)}$

2. Take a full step in time to update the position variable

$x_i(t + \delta) = x_i(t) + \delta\frac{\partial K}{\partial p_i(t+\delta/2)}$

3. Take the remaining half step in time to finish updating the momentum variable

$p_i(t + \delta) = p_i(t + \delta/2) - (\delta/2)\frac{\partial U}{\partial x_i(t+\delta)}$

This method can be run for L steps to simulate dynamics over L x δ units of time. This particular method preserves the energy of the system and is time-reversible, and thus it is a popular choice for numerical approximations, especially in comparison with Euler's method:

PUT PLOT OF EULER VS LEAPFROG FROM NEAL PAPER!!

4 MCMC from Hamiltonian dynamics

Using Hamiltonian dynamics to sample from a distribution requires translating the density function for this distribution to a potential energy function and introducing "momentum" variables to go with the original variables of interest (now seen as "position" variables). We can then simulate a Markov chain in which each iteration resamples the momentum and then does a Metropolis update with a proposal found using Hamiltonian dynamics.
How can we choose this Hamiltonian function? It turns out to be simple to relate $H(x, p)$ to $p(x)$ using a concept known as the **canonical distribution**. For any energy function $E(\theta)$, we define the corresponding canonical distribution as:

$p(\theta) = \frac{1}{Z}e^{-E(\theta)}$

The variable Z is a normalizing constant called the partition function that scales the canonical distribution such that is sums to one, creating a valid probability distribution.
Using the above equation for the total energy, we have:

$p(x, p) \propto e^{-H(x,p)}$
$= e^{-[U(x)-K(p)]}$
$= e^{-U(x)}e^{-K(p)}$
$\propto p(x)p(p)$

From above, see that the canonical distribution for x and p factorizes. We can now use Hamiltonian dynamics to sample from the joint canonical distribution over p and x and simply ignore the momentum contributions. Note that this is an example of introducing *auxiliary variables* to facilitate the Markov chain path.

4.1 Probability and the Hamiltonian

In HMC, we use Hamiltonian dynamics as a proposal function in order to explore the canonical (posterior) density $p(x)$ defined by $U(x)$ more efficiently. Starting at an initial state $[x_0, p_0]$, we simulate Hamiltonian dynamics for a short time using the Leap Frog method. We then use the position and momentum ariables at the end of the simulation as our porposed states variables x^* and p^*. This proposed state is accepted with an update rule similar to Metropolis before:

$p(x^*, p^*) \propto e^{-[U(x^*)+K(p^*)]}$
$p(x_0, p_0) \propto e^{-[U(x^{(t-1)}),K(p^{(t-1)})]}$

Accept with probability:

$\min(1, \frac{p(x^*,p^*)}{p(x^0,p^0)})$

Similar to the Metropolis algorithm, if the state is rejected, the next state of the Markov Chain is set at the state (t-1). Note that Hamiltonian Dynamics will follow contours of constant energy in phase space. Therefore, in order to explore all of the posterior distribution, we draw a random momentum from the corresponding canonical distribution $p(\mathbf{p})$ before running the dynamics prior to each sampling iteration t.

4.2 Hamiltonian Monte Carlo Algorithm

1. set t = 0
2. generate an initial position state $x^{(0)} \sim \pi^{(0)}$
3. repeat until t = M
   a. set t = t+1
      i. sample a new initial momentum variable from the momentum canonical distribution $p_0 \sim p(p)$
      ii. set $x_0 = x^{(t-1)}$
      iii. run Leap Frog algorithm starting at $[x_0, p_0]$ for L steps and stepsize δ to obtain proposed states x^* and p^*
      iv. calculate the Metropolis acceptance probability:
      - $\alpha = \min(1, \exp(-U(x^*) + U(x_0) - K(p^*) + K(p_0)))$
      v. draw a random number u from Unif(0,1)
      - if $u \leq \alpha$ accept the proposed state position x^* and set the next state in the Markov chain $x^{(t)} = x^*$
      - else set $x^{(t)} = x^{(t-1)}$

5 Empirical Comparisons

5.1 Data set

5.2 Comparisons against other MCMC methods

6 Conclusion

7 References